

FRAMEWORK HAVING PLUG-AND-PLAY FUNCTION AND ITS RECONSTRUCTING METHOD

Publication number: JP2001290724

Publication date: 2001-10-19

Inventor: VISHAL SHAH

Applicant: NIPPON ELECTRIC CO

Classification:

- international: G06F13/38; G06F13/00; G06F15/00; H04L12/28;
H04L29/06; G06F13/38; G06F13/00; G06F15/00;
H04L12/28; H04L29/06; (IPC1-7): G06F13/00;
G06F13/38; G06F15/00; H04L12/28; H04L29/06

- European:

Application number: JP20000371402 20001206

Priority number(s): US20000546397 20000410

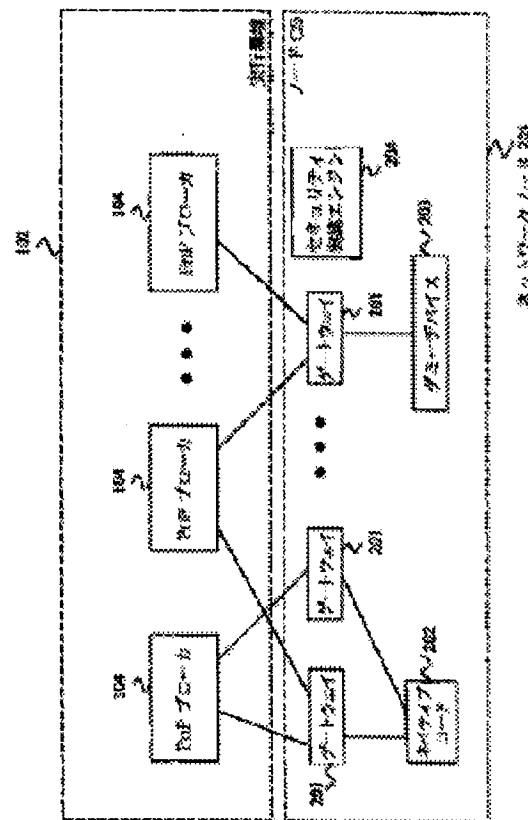
Also published as:

 JP2004334896 (A)

Report a data error here

Abstract of JP2001290724

PROBLEM TO BE SOLVED: To enable a user application to find and utilize various network devices by integrating a plurality of networks having different communication protocols into a single unified framework. **SOLUTION:** This active configuration framework system interconnects a service and a service user. The framework has a plug-and-play(PnP) broker 104. The service user finds a networking service by using the PnP broker, utilizes the networking service and communicates with the networking service. The service user communicates with the service through the interface of the PnP broker independently of a communication protocol used by a device offering the service. A gateway 201 for registering the service in the PnP broker is also provided.



Data supplied from the **esp@cenet** database - Worldwide

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2001-290724

(P2001-290724A)

(43) 公開日 平成13年10月19日 (2001. 10. 19)

(51) Int.Cl. ⁷	識別記号	F I	特コード* (参考)
G 0 6 F 13/00	3 5 7	G 0 6 F 13/00	3 5 7 A
	3 5 1		3 5 1 B
13/38	3 5 0	13/38	3 5 0
15/00	3 1 0	15/00	3 1 0 D
H 0 4 L 12/28		H 0 4 L 11/00	3 1 0 Z

審査請求 有 請求項の数32 O L (全 24 頁) 最終頁に続く

(21) 出願番号 特願2000-371402(P2000-371402)

(22) 出願日 平成12年12月6日(2000. 12. 6)

(31) 優先権主張番号 09/546397

(32) 優先日 平成12年4月10日(2000. 4. 10)

(33) 優先権主張国 米国 (US)

(71) 出願人 000004237

日本電気株式会社

東京都港区芝五丁目7番1号

(72) 発明者 ビシヤル・シャー

アメリカ合衆国、カリフォルニア 95134

サンノゼ、110 ロブルスアベニュー、

エヌ・イー・シー・ユー・エス・エー・イ
ンク内

(74) 代理人 100097157

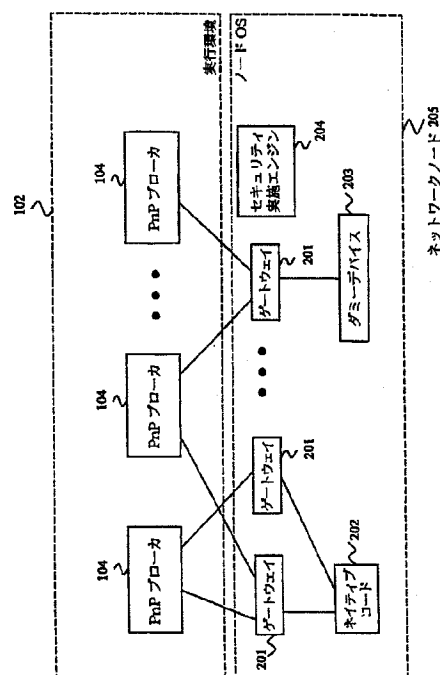
弁理士 桂木 雄二

(54) 【発明の名称】 プラグアンドプレイ機能を有するフレームワークおよびその再構成方法

(57) 【要約】

【課題】 相異なる通信プロトコルを有する複数のネットワークを単一の統一フレームワークへと統合して、ユーザアプリケーションがさまざまなネットワークデバイスを発見し利用することを可能にする。

【解決手段】 本発明のアクティブコンフィグレーションフレームワークシステムは、サービスとサービスユーザを相互接続する。このフレームワークは、プラグアンドプレイ (PnP) ブローカ 104 を有する。サービスユーザは、PnP ブローカを用いて、ネットワークサービスを発見し、利用し、ネットワークサービスと通信する。サービスユーザは、サービスを提供するデバイスによって使用される通信プロトコルとは独立に、PnP ブローカのインタフェースを通じてサービスと通信する。また、サービスを PnP ブローカに登録するゲートウェイ 201 が提供される。



【特許請求の範囲】

【請求項1】 (a) 少なくとも1つのサービスおよび少なくとも1つのサービスユーザと、

(b) 第1のプラグアンドプレイブローカとを有するアクティブコンフィグレーションフレームワークにおいて、

前記少なくとも1つのサービスユーザは、前記第1のプラグアンドプレイブローカを用いて、前記サービスを利用することを特徴とするアクティブコンフィグレーションフレームワーク。

【請求項2】 前記サービスユーザは、前記サービスを提供するデバイスによって使用される通信プロトコルとは独立に、前記第1のプラグアンドプレイブローカを通じて前記サービスと通信することを特徴とする請求項1に記載のアクティブコンフィグレーションフレームワーク。

【請求項3】 前記サービスは、ゲートウェイおよびデバイスクラスタを有し、

前記ゲートウェイは、前記デバイスクラスタを前記第1のプラグアンドプレイブローカに接続し、

前記第1のプラグアンドプレイブローカは、前記ゲートウェイを通じて、前記デバイスクラスタと通信することを特徴とする請求項1に記載のアクティブコンフィグレーションフレームワーク。

【請求項4】 前記第1のプラグアンドプレイブローカは、第1の通信プロトコルを用いて前記ゲートウェイに接続され、

前記ゲートウェイは、異なる第2の通信プロトコルを用いて前記デバイスクラスタに接続され、

前記ゲートウェイは、前記第1の通信プロトコルと前記第2の通信プロトコルの間の変換を行うことを特徴とする請求項3に記載のアクティブコンフィグレーションフレームワーク。

【請求項5】 前記ゲートウェイは、前記第1のプラグアンドプレイブローカ内の、前記サービスに対応する属性を設定することによって、前記サービスを前記第1のプラグアンドプレイブローカに登録することを特徴とする請求項3に記載のアクティブコンフィグレーションフレームワーク。

【請求項6】 前記ゲートウェイは、前記第1のプラグアンドプレイブローカからの要求に回答して、前記第1のプラグアンドプレイブローカで前記サービスのステータスを周期的に更新することを特徴とする請求項3に記載のアクティブコンフィグレーションフレームワーク。

【請求項7】 前記サービスユーザは、前記第1のプラグアンドプレイブローカから離れたリモートサービスユーザであり、

前記リモートサービスユーザは、インターネット通信プロトコルを通じて前記第1のプラグアンドプレイブローカと通信することを特徴とする請求項1に記載のアク

ティブコンフィグレーションフレームワーク。

【請求項8】 前記アクティブコンフィグレーションフレームワークは、第2のプラグアンドプレイブローカをさらに有し、

前記サービスユーザは、前記第2のプラグアンドプレイブローカと通信し、

前記第2のプラグアンドプレイブローカは、前記第1のプラグアンドプレイブローカと通信して、前記少なくとも1つのサービスユーザが前記サービスを利用することを可能にすることを特徴とする請求項1に記載のアクティブコンフィグレーションフレームワーク。

【請求項9】 前記アクティブコンフィグレーションフレームワークは、ディレクトリエージェントをさらに有し、

前記第1のプラグアンドプレイブローカは、サービスロケーションプロトコルを用いて、前記サービスを前記ディレクトリエージェントに登録することを特徴とする請求項8に記載のアクティブコンフィグレーションフレームワーク。

20 【請求項10】 前記第1のプラグアンドプレイブローカは、サービスレジストリを有し、

前記サービスレジストリは、前記サービスおよび他のネットワークサービスに関する情報を含み、

前記少なくとも1つのサービスユーザは、前記サービスレジストリを用いて、前記サービスおよび前記他のネットワークサービスを発見することを特徴とする請求項1に記載のアクティブコンフィグレーションフレームワーク。

30 【請求項11】 前記アクティブコンフィグレーションフレームワークは、第2のプラグアンドプレイブローカをさらに有し、

前記第2のプラグアンドプレイブローカは、前記第1のプラグアンドプレイブローカに問合せを送ることによって、前記サービスを発見することを特徴とする請求項1に記載のアクティブコンフィグレーションフレームワーク。

【請求項12】 前記アクティブコンフィグレーションフレームワークは、第2のプラグアンドプレイブローカおよびディレクトリエージェントをさらに有し、

40 (a) 前記少なくとも1つのサービスユーザは、要求されるサービスの記述を前記第2のプラグアンドプレイブローカに送り、

(b) 前記第2のプラグアンドプレイブローカは、前記ディレクトリエージェントと通信し、

(c) 前記ディレクトリエージェントは、前記要求されるサービスの記述に一致するサービスのアドレスを前記第2のプラグアンドプレイブローカに返し、

50 (d) 前記第2のプラグアンドプレイブローカは、前記アドレスを用いて、前記サービスユーザと前記サービスの間にセッションを確立することを特徴とする請求項1

に記載のアクティブコンフィグレーションフレームワーク。

【請求項13】 (1) 前記サービスは、ゲートウェイおよびデバイスクラスタを有し、

(2) 前記ゲートウェイは、前記デバイスクラスタを前記第1のプラグアンドプレイブローカに接続し、

(3) 前記第1のプラグアンドプレイブローカは、前記ゲートウェイを通じて、前記デバイスクラスタと通信し、

(4) 前記ゲートウェイは、前記第1のプラグアンドプレイブローカ内の、前記サービスに対応する属性を設定することによって、前記サービスを前記第1のプラグアンドプレイブローカに登録し、

(5) 前記第1のプラグアンドプレイブローカは、前記サービスを前記ディレクトリエージェントに登録することを特徴とする請求項10に記載のアクティブコンフィグレーションフレームワーク。

【請求項14】 前記第1のプラグアンドプレイブローカは、

(a) 前記サービスユーザと通信するプラグアンドプレイブローカインタフェースと、

(b) 前記第1のプラグアンドプレイブローカと、前記フレームワーク内の他のプラグアンドプレイブローカとの間の通信のためのプラグアンドプレイブローカ間プロトコルと、

(c) 前記サービスに関する情報を格納する少なくとも1つのサービスレコードを有するサービスレジストリと、

(d) 前記サービスのネットワークアドレスを探索し前記サービスのアベイラビリティステータスを取得するサービスディスカバリ／アベイラビリティエージェントと、

(e) 前記サービスと前記サービスユーザの間の通信セッションを確立するサービスセッション管理エージェントと、

(f) 前記サービスのアドレスを探索するサービスローケーションプロトコルユーザエージェントと、

(g) 前記サービスを公表するサービスローケーションプロトコルサービスエージェントとを有することを特徴とする請求項1に記載のアクティブコンフィグレーションフレームワーク。

【請求項15】 前記少なくとも1つのサービスレコードは、前記サービスのタイプおよび複数のサービス属性レコードを格納するサービス識別フィールドを有し、前記サービス属性レコードは、前記サービス属性のタイプを格納する属性識別フィールドと、前記サービス属性の値を格納する属性値フィールドと、前記サービスのアクセス制御情報を格納するアクセス制御フィールドとを有することを特徴とする請求項14に記載のアクティブコンフィグレーションフレームワーク。

【請求項16】 (a) 少なくとも1つのサービスと、
(b) サービスエージェントと、

(c) ディレクトリエージェントとを有するアクティブコンフィグレーションフレームワークにおいて、

前記サービスエージェントは、前記少なくとも1つのサービスのタイプおよび属性を前記ディレクトリエージェントに登録し、

前記ディレクトリエージェントは、ユーザアプリケーションによって要求されるサービスのタイプおよび属性を、前記少なくとも1つのサービスのタイプおよび属性と照合して、前記少なくとも1つのサービスのアドレスを返すことを特徴とするアクティブコンフィグレーションフレームワーク。

【請求項17】 前記アクティブコンフィグレーションフレームワークは、ユーザエージェントをさらに有し、前記ユーザエージェントは、前記要求されるサービスのタイプおよび属性を前記ディレクトリエージェントに送信し、

前記ディレクトリエージェントは、前記少なくとも1つのサービスのアドレスを前記ユーザエージェントに返すことを特徴とする請求項16に記載のアクティブコンフィグレーションフレームワーク。

【請求項18】 前記少なくとも1つのサービスは、ゲートウェイおよびデバイスクラスタを有し、

前記ゲートウェイは、前記デバイスクラスタを前記サービスエージェントに接続し、

前記サービスエージェントは、前記ゲートウェイを通じて、前記デバイスクラスタと通信することを特徴とする請求項16に記載のアクティブコンフィグレーションフレームワーク。

【請求項19】 前記アクティブコンフィグレーションフレームワークは、実行環境をさらに有し、前記アプリケーションは、前記実行環境で動作することを特徴とする請求項16に記載のアクティブコンフィグレーションフレームワーク。

【請求項20】 前記実行環境はJava仮想マシンであることを特徴とする請求項19に記載のアクティブコンフィグレーションフレームワーク。

【請求項21】 前記実行環境はWindows™ベースのマシンであることを特徴とする請求項19に記載のアクティブコンフィグレーションフレームワーク。

【請求項22】 ネットワークデバイスの追加にตอบสนองしてフレームワークの自動再構成を行う方法において、前記ネットワークデバイスは、少なくとも1つのサービスを提供し、

前記フレームワークは、サービスエージェントおよびディレクトリエージェントを有し、

前記方法は、

(a) 前記サービスエージェントが、提供されるサービスのタイプおよび属性を前記ディレクトリエージェント

に登録するステップと、

(b) 前記ディレクトリエージェントが、ユーザアプリケーションによって要求されるタイプおよび属性を、前記提供されるサービスのタイプおよび属性と照合して、前記提供されるサービスのアドレスを返すステップと、を有することを特徴とする、ネットワークデバイスの追加にตอบสนองしてフレームワークの自動再構成を行う方法。

【請求項 23】 前記フレームワークは、ユーザエージェントをさらに有し、

前記ユーザエージェントは、前記要求されるサービスのタイプおよび属性を前記ディレクトリエージェントに送信し、

前記ディレクトリエージェントは、前記提供されるサービスのアドレスを前記ユーザエージェントに返すことを特徴とする請求項 22 に記載の方法。

【請求項 24】 前記ステップ (a) は、

(1) 前記ゲートウェイが、前記ネットワークデバイスのアベイラビリティをチェックするステップと、

(2) 前記ゲートウェイが、前記提供されるサービスのタイプおよび属性を前記サービスエージェントに登録するステップと、

(3) 前記サービスエージェントが、前記提供されるサービスのタイプおよび属性を前記ディレクトリエージェントに登録するステップと、

を有することを特徴とする請求項 22 に記載の方法。

【請求項 25】 (c) セッション管理エージェントが、前記アドレスを用いて、前記ユーザアプリケーションと前記ネットワークデバイスの間の通信セッションを確立するステップをさらに有することを特徴とする請求項 22 に記載の方法。

【請求項 26】 フレームワークに挿入されたデバイスによって実行されるサービスのための通信インタフェースを生成する方法において、

前記インタフェースは、ユーザアプリケーションによって、前記サービスを利用するために使用され、

前記方法は、

(a) XML 文書、文書型定義、またはスタイルシートのうちの少なくとも 1 つを含む、前記デバイスによって提供されるサービスのサービス記述スキーマを生成するステップと、

(b) XML パーサを用いて、前記サービス記述スキーマから、前記サービスのための通信インタフェースを生成するステップと、を有することを特徴とする、フレームワークに挿入されたデバイスによって実行されるサービスのための通信インタフェースを生成する方法。

【請求項 27】 前記サービス記述スキーマは、前記サービスのためのデータおよび制御フロー仕様を含むことを特徴とする請求項 26 に記載の方法。

【請求項 28】 前記ステップ (b) において、前記デ

バイスによって生成されるサービス記述スキーマは、ユーザインタフェースウィジェットのセットにマッピングされ、

ユーザは、実行のために前記デバイスにコマンドを送るユーザインタフェースウィジェットをクリックすることによって、前記デバイスの状態を変えることができることを特徴とする請求項 26 に記載の方法。

【請求項 29】 前記ステップ (b) において、前記デバイスによって生成されるサービス記述スキーマは、前記サービスのための通信インタフェースを表すオブジェクトの階層にマッピングされ、

前記サービスは、前記オブジェクトの階層を用いて、前記ユーザアプリケーションによって利用されることを特徴とする請求項 26 に記載の方法。

【請求項 30】 前記オブジェクトの階層の構造は、前記ユーザアプリケーションが前記デバイスを利用する時点での前記デバイスの機能に基づいて決定されることを特徴とする請求項 29 に記載の方法。

【請求項 31】 前記 XML パーサは、XML 対応ウェブブラウザであることを特徴とする請求項 30 に記載の方法。

【請求項 32】 フレームワークに挿入されたデバイスを制御する方法において、該方法は、

(a) 前記デバイスが、XML 文書、文書型定義、またはスタイルシートのうちの少なくとも 1 つを含む、前記デバイスのデバイス記述スキーマを生成するステップと、

(b) 前記ユーザアプリケーションが、前記デバイス記述スキーマを編集し、編集されたデバイス記述スキーマを前記デバイスに反映するステップとを有し、前記デバイスは、前記編集されたデバイス記述スキーマに従って状態を変更することを特徴とする、フレームワークに挿入されたデバイスを制御する方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、相異なる通信プロトコルを有する複数のネットワークを単一の統一フレームワークへと統合して、ユーザアプリケーションがさまざまなネットワークデバイスを発見し利用することを可能にする方法および装置に関する。また、本発明は、利用可能な機能の変更およびユーザ相互作用にตอบสนองしてネットワークノードの動的な適応および再構成を達成する方法に関する。また、本発明は、アプリケーション、サービス、およびデバイスが、自己の能力を記述し、それを他のアプリケーション、サービス、およびデバイスに公表する方法に関する。さらに、本発明は、物理的に相異なるデバイスが、接続し、情報交換し、データタイプを交渉し、それぞれの動作についてのステータスを提供して、ネットワークプラグアンドプレイを達成することを可能にする、プラットフォーム独立でトランスポート

独立なプロトコルを提供する方法に関する。

【0002】

【従来の技術】世界的なネットワーキング基盤の出現により、分散サービスおよび分散アプリケーションの大規模な配備が一般的になっている。現在および将来のネットワークでは、時間に敏感な情報を全世界に発信し、世界的な取引を電子的に仲介するサービスがさらに配備されることが期待される。このような分散サービスが普遍的になる前に、このようなサービスの開発、デバッグ、配備および発展を簡単にする新たな機構が必要とされる。このような機構は、基礎になるプロトコルやインタフェースにかかわらずに、必要なサービスを発見し、そのサービスの能力を使用することが必要となる。

【0003】家庭で使用可能なもののような典型的な分散ネットワークシステムは、相異なるさまざまな機器を相互接続する。さまざまな家庭用機器（ホームデバイス）は本質的にさまざまなインタフェースおよびプロトコルを利用することがあるため、単一の共通の通信プロトコルを用いてこれらのすべてのデバイスを相互接続することはほとんど不可能である。場合によっては、ほとんどの家庭に既に設置されている配線を使用する（例えば、既存の電源線をX-10デバイスとの通信に使用する）ことは便利であるが、このような既存の配線を使用するネットワークの伝送速度が制限される。例えば、白熱電球やホームセキュリティシステムのようなX-10デバイスは、毎秒数ビットのデータレートで動作する。他方、デジタルTV、カメラ、あるいはVCRのようなデバイスは、ずっと高い伝送データレートを必要とする。このようなデバイスは通常、IEEE1394アーキテクチャプロトコルを用いて相互接続され、データ転送レートは400Mb/sに達する。

【0004】これに対して、現在開発されているさまざまな家庭用自動化アプリケーションは、さまざまな家庭用機器の有効な相互作用を必要とする。したがって、家庭に設置されたすべての相異なるデバイスおよびサービスの有効な相互作用を可能にするには、相異なるさまざまなネットワークを、統一するフレームワークに統合して、相異なるネットワーク内に位置するさまざまなネットワークエンティティが互いを発見し相互作用する機構を提供しなければならない。

【0005】また、さまざまなデバイスインタフェースおよびネットワークプロトコルについて、低レベルのプロトコル固有の詳細をアプリケーションが扱うことを必要とせずに、リモートサービスを利用する手段をユーザアプリケーションに提供することが望ましい。

【0006】真のプラグアンドプレイ能力を達成するには、フレームワークは、新たに追加されたデバイスによって提供されるサービスを自動的に発見する機構を提供しなければならない。このようなフレームワーク内では、利用可能なネットワークサービスおよびそのネット

ワーク位置に関する記述は、すべてのユーザアプリケーションにとって容易に利用可能であるようにしなければならない。さらに、ユーザアプリケーションは、遭遇する可能性のある特定のデバイスあるいはサービスとの通信のために、自己の通信インタフェースを自動的に再構成することができなければならない。さらに、フレームワークは、ネットワークサービスへのアクセスを制御するために、ユーザの認可および認証を実行するセキュリティモデルを含まなければならない。

10 【0007】Jini™は、ネットワークプラグアンドプレイを達成する既存の候補のうちの1つである（Jiniは、サンマイクロシステムズ社(Sun Microsystems, Inc.)の商標である）。Jiniは、分散システムの構築および配備を容易にするアプリケーションプログラミングインタフェース(API)およびランタイム規約のセットである。Jiniは、分散システムの、共通しているが相異なる部分処理する「配管」(plumbing)を提供する。Jiniは、プログラミングモデルおよびランタイムインフラストラクチャからなる。リース、分散イベント、および分散トランザクションをサポートするAPIおよび規約を定義することによって、プログラミングモデルは、基礎となるネットワークの信頼性が低くても、信頼性の高い分散システムを開発者が構築するのを助ける。ランタイムインフラストラクチャは、ネットワークプロトコルとそれを実装するAPIとからなり、ネットワーク上のデバイスおよびサービスの追加、探索、アクセス、および削除を容易にする。

20 【0008】Jiniの使用は、基礎となるネットワーク技術について知らなくてもサービスを発見する容易な方法を提供する。他の既知のインフラストラクチャに比べてJiniが改良されている点は、ユーザアプリケーションが実際にサービスを探索し、そのサービスをサポートするホストを他者に発見させることができることである。しかし、主な未解決の問題点は、「ユーザアプリケーションはどのようにして自動的にクライアントマシン上のサービスを使用するか」である。クライアントユーザアプリケーションにサービスインタフェースしか設けられていない場合、クライアントが、このサービスを理解することが可能なコードを有していなければならない。

40 【0009】例えば、ユーザが、自己のアプリケーションに、カメラサービスのためのネットワークをブラウズさせたい状況を考える。アプリケーションがこのようなサービスをネットワーク上に見つけた場合、ユーザは、それをクリックして、ユーザ自身のアプリケーションでそれを使用することができる。ユーザが、このようなサービスを実行するためのコードを必要とする場合、このようなコードは、ユーザのマシン上に自動的にダウンロードされなければならない。しかし、このアプローチに
50 関連する主な問題点は、ユーザアプリケーションがこの

ようなサービスをどのようにして利用するかである。ユーザアプリケーションは、拡張可能なインタフェースを有する場合、リモートサービスインタフェースに問い合わせることでその固有の特性を取得し、その利用を可能にすることができる。しかし、このような拡張可能インタフェースの定義は、Jini™仕様の範囲内ではなく、Jini™サービスを使用するユーザアプリケーションに委ねられている。

【0010】もう1つの問題点は、カメラサービスのセマンティクス（意味論）を定義するエンティティの識別である。例えば、キャノン、ミノルタ、およびニコンを含む主要なカメラメーカーがそれぞれ固有のインタフェースを提供する場合、カメラサービスには相互運用性がなくなり、ユーザは、3つの異なるすべてのカメラサービスをコードの形で適切に提供しなければならないことになる。また、もしコダックがJini™カメラを市販することに決めた場合、ユーザは、コダックのインタフェースの解析を含むようにそのコードを手動（マニュアル）で変更しなければならない。このような拡張可能インタフェースは、ネットワークトラフィックと、クライアントコードのサイズを非常に増大させてしまう。すべての種類の市場固有のグループがまとまってインタフェースを指定することが好ましく、多くの場合にはこれが必要となるであろうが、これはすぐには、また効率的には実現しそうになく、特に、家庭で使用されるすべての種類のデバイスに対しては、きわめて非現実的である。

【0011】既知のネットワークで依然として解決されていないもう1つの制限は、Jini™ベースのアプリケーションがどのようにして非Jini™ベースのオペレーティングシステム上にあるデータにアクセスすることができるかに関するものである。すなわち、ユーザは、例えば、Jini™に対するマイクロソフトのサポートなしで、どのようにしてMS-Word™からJini™プリンタにアクセスすることができるだろうか。Jini™プリンタを使用したいJava™アプリケーションですら、アクセスのためのオペレーティングシステムに依拠しなければならない。

【0012】例えば、ユーザプログラムが印刷メソッドへのアクセスを要求するとき、以下のようなコードフラグメントが使用される。

```
object o=Lookup(GeneralPrintService);
GeneralPrintService pservice=(GeneralPrintService)
o;
pservice.print();
```

【0013】しかし、このコードが書かれるときに、プログラムは、GeneralPrintServiceという名前のインタフェースがコード実行時に本当に存在するかどうかを知らない。さらに、プログラムが複数のデバイスでイメージを印刷したい場合、複数のインタフェースを予想し、それに応じて印刷を行う必要がある。

【0014】Jiniは、ユーザインタフェース（UI）コンポーネントがサービスの属性リストにエントリとして付加されるように使用することができるため、サービスについて要求されるクライアントの知識は最小限である。クライアントは、単に、UIがアプレットと同様の属性によって表現されることを知っているだけでよい。UIは、JavaBeansとして提供可能であるため、内省（introspection）・反省（reflection）により、サービスは、クライアントに対して、クライアントがそのサービスを使用することができる手段を提供することが可能である。これは、本質的に、クライアントとサーバの間のメッセージ交換であり、手動の設定を必要とする。しかし、UIコンポーネントを付加することは、人間でないクライアント（生成・使用されるマシン）がデバイスを使用するのを妨げる可能性がある。また、デバイスの個数が増大して、クライアントプログラムが複雑になると、もう1つの問題点が生じる。その場合、クライアントは、返されたリストのどのオブジェクトが真のターゲットであるかを決定しなければならない。さらに、あらゆるJini™のデバイスあるいはアプリケーションは、製造時において通信するために必要としたすべてのデバイスの（あるいは少なくとも、「デバイスのすべてのタイプの」）すべてのインタフェースを必要とする。その結果、Jini™は、相互運用性の問題点を実際には解決していない。

【0015】もう1つの既存のネットワークシステムであるHomeAPIは、アプリケーションがさまざまなホームデバイスを発見し利用することを可能にする、オブジェクト指向のソフトウェアサービスおよびアプリケーションプログラミングインタフェース（API）のセットである。このシステムは、アプリケーションが使用するデバイスおよびネットワークのプロトコル固有の詳細に、ユーザアプリケーションが関与しないように設計された。HomeAPIは、HomeAPIインタフェースの使用を通じてサービスを利用するユーザアプリケーションを開発するのを容易にする分散サービスを表現するソフトウェアオブジェクトのセットを提供する。

【0016】しかし、HomeAPIは、新たなデバイスの発見の問題点を解決していないため、所望のプラグアンドプレイネットワーク能力を提供しない。HomeAPIのもう1つの欠点は、すべてのネットワークデバイスが標準の通信インタフェースを提供することを要求することであり、これは今日の市場では実現不可能である。

【0017】HomePNA、HomeRF、Bluetooth、PIANO、X-10、CEBus、IEEE1394、USB、HAViなどのようなその他の発展段階のネットワーク構想は、単に、ネットワーク接続に関するさまざまなプロトコルを提供しているだけで

ある。これらの通信プロトコルは当業者に周知である。しかし、現在のところ、競合するさまざまなネットワーク技術およびプロトコルの相互運用性の問題点を解決し、ネットワークプラグアンドプレイを達成する統一フレームワークシステムは存在しない。

【0018】

【発明が解決しようとする課題】したがって、さまざまなタイプのネットワークデバイスを相互接続し、デバイスの発見および利用のための一様な機構を提供する統一フレームワークに対して、強く幅広い受容が認識されており、このようなものがあることは非常に有利である。このような統一フレームワークは、ユーザアプリケーションが、アプリケーション-サービス間通信プロトコル、所望のサービスを提供するデバイスに接続されたホストのIPアドレスの発見、およびさらには、ネットワークサービスのユーザインタフェースの詳細のような低レベルの詳細を組み込むことを不要にするとともに、上記のその他の要求を満たす。

【0019】

【課題を解決するための手段】したがって、本発明の1つの目的は、既知のネットワークシステムの上記の制限を克服し、さまざまな機器を相互接続する相異なる通信プロトコルを有する複数のネットワークを単一の統一フレームワークへと統合して、ユーザアプリケーションがさまざまなネットワークデバイスを発見し利用することを可能にする方法および装置を提供することである。

【0020】具体的には、本発明の目的は、利用可能な機能の変更およびユーザ相互作用にตอบสนองしてネットワークノードの動的な適応および再構成を達成する方法および装置を提供することである。

【0021】本発明のもう1つの目的は、アプリケーション、サービス、およびデバイスが、自己の能力および通信インタフェースを記述し、それを他のアプリケーション、サービス、およびデバイスに公表する方法を提供する。

【0022】本発明のさらにもう1つの目的は、物理的に相異なるデバイスが、接続し、情報交換し、データタイプを交渉し、それぞれの動作についてのステータスを提供して、ネットワークプラグアンドプレイを達成することを可能にする、プラットフォーム独立でトランスポート独立なプロトコルを提供する方法および装置を提供することである。

【0023】上記の目的を実現し、本発明の効果を達成するため、アクティブコンフィグレーションフレームワークシステムが提供される。本発明のフレームワークは、サービスとサービスユーザを相互接続する。また、このフレームワークは、プラグアンドプレイブローカを有する。本発明のフレームワークでは、サービスユーザは、プラグアンドプレイブローカを用いて、ネットワー

キングサービスを発見し、利用し、ネットワークングサービスと通信する。

【0024】本発明のフレームワークでは、サービスユーザは、サービスを提供するデバイスによって使用される通信プロトコルとは独立に、プラグアンドプレイブローカのインタフェースを通じてサービスと通信する。

【0025】また、本発明によれば、サービスをプラグアンドプレイブローカに登録するゲートウェイが提供される。

【0026】また、本発明によれば、サービスエージェント、ユーザエージェント、およびディレクトリエージェントを有するアクティブコンフィグレーションフレームワークが提供される。本発明のフレームワークでは、サービスエージェントはサービスをディレクトリエージェントに登録する。その後、ユーザエージェントは、要求するサービスの記述をディレクトリエージェントに通信し、ディレクトリエージェントは、適合するサービスのロケーションを帰す。

【0027】本発明のフレームワークでは、サービスは、デバイスクラスおよびゲートウェイからなることが可能である。サービスエージェントは、このゲートウェイを通じてデバイスクラスと通信する。

【0028】また、本発明によれば、新たなネットワークデバイスの追加にตอบสนองして、フレームワークの自動的再構成を行う方法が提供される。本発明の方法によれば、サービスエージェントは、サービスをディレクトリエージェントに登録する。その後、ユーザエージェントは、要求するサービスの記述をディレクトリエージェントに通信する。最後に、ディレクトリエージェントは、要求されたサービスの記述を、提供するサービスの記述と照合し、適合したサービスのアドレスをユーザエージェントに返す。

【0029】また、本発明によれば、フレームワークに追加される新たなネットワークデバイスによって実行されるサービスに対する通信インタフェースを生成する方法が提供される。本発明の方法によれば、デバイスは、その通信インタフェースを記述するスキーマを生成する。このスキーマは、XML (eXtensible Markup Language) 文書の形式とすることが可能である。その場合、ユーザアプリケーションは、XML言語パーサを用いて、通信インタフェースを生成する。

【0030】また、本発明によれば、ネットワークデバイスを制御する方法が提供される。本発明の方法によれば、ユーザアプリケーションは、デバイス記述スキーマを編集し、編集されたデバイス記述スキーマをデバイスに返す。これにตอบสนองして、デバイスは、編集されたデバイス記述スキーマに従って状態を変更する。

【0031】

【発明の実施の形態】すべてのタイプの分散デバイスおよび分散サービスにわたりネットワークプラグアンドブ

レイを達成するためには、統一フレームワークが必要である。ここで使用される「フレームワーク」あるいは「サブストレート」という用語は、1つ以上のネットワークを統合するシステムを意味する。しかし、多くの場合、「フレームワーク」、「サブストレート」、および「ネットワーク」という用語は区別なく用いられる。

【0032】本発明のフレームワークを使用することにより、ネットワーク内の相異なる物理デバイスの個数にかかわらず、ネットワークエンティティの相互作用のための複雑な機構は一度だけ実装すればよい。さらに、これらの機構は、デバイス固有のゲートウェイを通じて管理ブローカと相互作用するユーザアプリケーションから隠蔽される。本発明のフレームワークは、SLP、HTTP、XML、Java RMI、およびIIOPのような周知のインターネット関連プロトコルに基づいており、ウェブサーバへのアドオンとして実装可能であるため、配備および利用が容易である。さらに、本発明のフレームワークでは、サービス発見の全プロセスは、ユーザアプリケーションにとって透過的（トランスペアレント）であり、物理接続（コネクション）に対する区別がない。例えば、ネットワークに接続されるプラグアンドプレイデバイスはTCP/IPベースである必要はない。サービス利用プロセスもまたトランスペアレントであり、対象となるさまざまなデバイスの能力と、ユーザの認証および認可に基づく。互換性のないデバイスを接続するときには、トンネリングや完全なデータ変換が必要となることもある。

【0033】本発明は、デバイス独立な情報交換を容易にするためのアクティブコンフィグレーションフレームワークを提供する。複数のデバイスが接続し、情報交換し、データタイプを交渉し、それぞれの動作についてのステータスを提供するには、相異なるプロトコルが要求される。これらのプロトコルは、プラットフォーム独立であり、その形式や機能にかかわらず、他のデバイスに接続する必要のある任意のデバイスに組み込むことができる。これらのプロトコルはトランスポート独立でもあるため、任意のデバイスが、他の任意のデバイスに接続し、情報交換することが可能となる。

【0034】本発明のフレームワークでは、アクティブネットワークの場合のように、ユーザ定義計算がネットワーク内に配置される。しかし、アクティブネットワークとは異なり、現在のインターネットアーキテクチャのすべてのルーティングおよびフォワーディングのセマンティクスは、計算環境をアプリケーション層に制限することにより保持される。このフレームワークを利用するネットワークデバイスおよびサービスは、ネットワークインフラストラクチャに深く組み込まれる場合でも、実際には、エンドシステム上のユーザアプリケーションによって作成され、設定され、動的に制御される。このようなアプリケーション定義エンティティは、ネットワー

ク内の任意のノードで実行され、個々のパケットは、ネットワークを通じて伝搬するとともに任意のアクションを実行するようにプログラムされることが可能である。

【0035】[アクティブコンフィグレーションフレームワークのコンポーネント] 完全なアクティブコンフィグレーションアーキテクチャは、ダミーデバイス、ゲートウェイデバイス、および、ネットワークに直接的または間接的に接続されたPnPブローカからなる。アクティブコンフィグレーションフレームワークのアーキテクチャを図1に示す。図1に示すように、ネットワークには1個以上の実行環境102がある。実行環境102は、例えば、インターネットのようなネットワーク101を用いて相互接続される。実行環境102は、ある意味で、ネットワークノードとして作用する。実行環境102は、当業者に周知のJava仮想マシンやWindowsTMベースのマシン内にあることが可能である。実行環境102は、ゲートウェイ（図示せず）を通じて非インテリジェント（すなわち、ダミー）デバイスに接続された1個以上のPnPブローカ104を有する。ユーザアプリケーション103は、PnPブローカ104を通じて、デバイスを発見し、利用し、デバイスと通信する。以下で、本発明のアクティブコンフィグレーションフレームワークシステムのさまざまなコンポーネントについて詳細に説明する。

【0036】[ダミーデバイス] ダミーデバイスは、TCP/IPプロトコルスタック、もしくはJava仮想マシン、またはその両方のないデバイスである。このようなデバイスは、ネットワークから直接にアクセス可能ではなく、媒介物としてゲートウェイを必要とする。このようなデバイスは、ネットワークにインストールされる前に、特定のコンフィグレーション選択を行う必要がある。例えば、それぞれのX-10デバイス（白熱電球など）には、電源線ネットワークに接続する前に手動でハウスコードおよびデバイスコードを割り当てなければならない。さらに、このようなデバイスは、提示するサービスのためのセキュリティやアクセス制御を実施することはない。このようなデバイスのほとんどは、メモリや処理能力を有しておらず、ネットワークの残りの部分に依然として接続する必要のあるレガシーデバイスとして特徴づけることができる。

【0037】[ゲートウェイデバイス] 通常のネットワークデバイスは、単一のネットワークタイプ（例えば、IPまたはIPX）により、ピアツーピア方式で相互作用することができる。これは、あるネットワークタイプのデバイスは、同じネットワークに物理的に接続された、同じネットワークタイプのデバイスとのみ通信することができることを意味する。第1のデバイスが、その第1のデバイスによってサポートされていないネットワーク上にある第2のデバイス、あるいは、第1のデバイスによってサポートされていないプロトコルを実行して

いる第2のデバイスに接続することを必要とする場合に問題が生じる。この問題を解決するには2つのアプローチがある。

【0038】第1のアプローチは、複数の通信スタックを単一のネットワークデバイスにロードすることである。これにより、デバイスは、適当な通信スタックを有する限り、任意のタイプのネットワークデバイスと相互作用することが可能となる。基礎となるトランスポートは異なるが、デバイスがプロトコルスタック全体を理解する場合には、このアプローチはうまくいく。

【0039】複数の相異なるデバイスを接続する問題を解決するため、本発明のフレームワークシステムはゲートウェイを使用する。ゲートウェイは、ネットワーク内に不可視的に存在し、あるネットワークタイプを別のネットワークタイプに変換するデバイスである。このアプローチは、デスティネーションデバイスがレガシーであり、各ソースデバイスがデスティネーションと通信するためにデスティネーションをモデル化しなければならない、という本発明の実施例で用いられる。これは、例えば、ファックスデバイスや電子メールデバイスの場合である。デバイスにレガシーネットワークと対話する能力を追加するのではなく、この機能はゲートウェイに置かれる。ゲートウェイは、通常、プログラム可能なプラットフォーム上にホスティングされる。

【0040】本発明のフレームワークでは、PnPブローカが、アクセスのためにゲートウェイを要求するレガシー（ダミー）デバイスとのセッションを確立したい場合、ゲートウェイは、PnPブローカから、そのデバイスの存在を隠蔽する。PnPブローカは、その代わりに、ゲートウェイとのセッションを作成し、ダミーデバイスのアドレスをゲートウェイに渡す。この時点以降、ゲートウェイは、セッションにおけるいずれかのチャネルを通じてデータを受け取ると、そのデータを、ゲートウェイ-ダミーリンクを通じてリモートデバイスへと中継する。本発明は、次の2つのタイプのゲートウェイを使用する。一般化ゲートウェイは、IPデバイスとの通信を行い、特殊化ゲートウェイは、ダミーを通じて非IPデバイスとの通信を行う。特殊化ゲートウェイは、ダミーデバイスをネットワークに公開するものであり、そのデバイスを制御・管理するためのプログラミングロジックを含む。さらに、特殊化ゲートウェイは、ダミーデバイスのセキュリティポリシーを実現する。1つのゲートウェイデバイスが複数のダミーデバイスを管理することが可能である。

【0041】[PnPブローカ] PnPブローカは、アプリケーション、サービス、およびデバイスのサービスブローカとして作用する。PnPブローカは、分散システムのためのInfobusの概念を拡張し、ネットワーク全体をSoftware Busとして扱う。Infobusシステムは当業者に周知である。詳細な説明は、Reaz Hoq

ue, "ConnectingJavaBeans with InfoBus", Wiley Computer Publishing, Nov. 98, にある。Software Busもまた当業者に周知であり、Brian Oki, Manfred Pfluegl, Alex Siegel, and Dale Skeen, "The Information Bus - an architecture for extensible distributed systems", Proceedings of the 14th Symposium on Operating Systems Principles, p.58-68, Asheville, North Carolina, December 1993, に記載されている。

【0042】PnPブローカにより、ネットワーク接続されたエンティティは、他のネットワーク接続エンティティ（ゲートウェイあるいはダミー）の能力を発見し利用することが可能となる。ゲートウェイは、その能力を、対応するPnPブローカに登録し、ダミーデバイスは、ゲートウェイを発見し、このゲートウェイを通じてPnPブローカと通信する。ネットワーク内のデバイスの各タイプ（例えば、X-10X、1394X、USB）ごとに特定のゲートウェイがあり、その一端でこのデバイスクラスに接続され、他端でPnPブローカに接続される。X-10、HAVi、IEEE-1394、およびUSBは周知の通信プロトコルである。2つの相異なるゲートウェイが互いに対話したい場合、通信のためにPnPブローカを使用することができる。例えば、あるゲートウェイがJiniクラスを代表する一方で、別のゲートウェイが、HAViを用いたIEEE 1394クラスターのゲートウェイとなることが可能である。本発明のフレームワークでは、ゲートウェイは、そのアーキテクチャ用に設計されたダミーを使用することにより、受動（非TCP/IP）デバイスと通信する。

【0043】図2に、アクティブコンフィグレーションフレームワークの全体設計を示す。実行環境102内で動作するPnPブローカ104は、ゲートウェイ201を通じて、ダミーデバイス203およびユーザアプリケーション202と通信する。ゲートウェイデバイス201は、ネットワークノード205上で、そのオペレーティングシステム環境内で動作する。ノード205は、ゲートウェイおよびサービスへのアクセスを制御するためにセキュリティ実施機構204を提供する。

【0044】ネットワーク接続されたエンティティは、サービスユニットと呼ばれる有意義な機能に再分割される。原理的には、アーキテクチャは、ユーザアプリケーションの観点から、1つの有意義な機能を1つのサービスユニットとして定義する。サービスユニットは、ユーザアプリケーション/サービス全体であることも可能であり、ユーザアプリケーション/サービスの一部であることも可能である。コマンド、応答、およびデータは、ユーザアプリケーションとサービスユニットの間、サービスユニットとサービスの間、あるいはサービスユニットと別のサービスユニットの間で交換することが可能である。サービスユニットは、その機能をPnPブローカに登録する。

【0045】PnPブローカにより、ネットワーク接続されたエンティティは、他のネットワーク接続エンティティの能力を発見し利用することが可能となる。ネットワーク接続エンティティは、サービスユーザ（ユーザアプリケーション）であることが可能である。ユーザアプリケーションは、PnPブローカを通じて、サービスを発見し、それを使用することを要求する。PnPブローカは、サービスユニットとユーザアプリケーションを接続するためのトランスポート独立なインタフェース（PnPブローカインタフェースという）を提供する。このインタフェースは、TCP/IPプロトコルスタック上で動作するように設計される。PnPブローカは、サービスブローカまたはマネージャとしてその役割を果たすために、異なるクラスタ内の他のPnPブローカと通信する。PnPブローカ間のプロトコルはPnPブローカ間プロトコルと呼ばれる。

【0046】各デバイスクラスタは、高々、1つのPnPブローカを有する。ローカルなPnPブローカがないとき、ユーザアプリケーション/サービスユニットは、Java RMIやHTTPのようなインターネット通信プロトコルを通じてリモートPnPブローカ（別のゲートウェイに接続されたPnPブローカ）を使用することが可能である。Java RMIプロトコルは当業者に周知である。その詳細な説明は、Downing, Troy Bryan, "Java RMI: Remote Method Invocation", IDG Books Worldwide, 1998, にある。HTTPプロトコルも当業者に周知である。その説明は、"Paul S. Hethmon, "Illustrated Guide to HTTP", Manning Publications, 1997, にある。サービスユニットは、それに直接に接続されたPnPブローカを使用することも可能であるし、PnPブローカ間プロトコルを使用することによりリモートPnPブローカを使用することも可能である。図3に示すネットワークシステムでは、サービスユニット406は、PnPブローカ間プロトコル404を通じて、リモートPnPブローカ1104に接続されたサービス1406および1402を利用する。こうして、あるサービスユニットが、異なるデバイスクラスタに位置するサービスユニットで利用可能なサービスに問い合わせ、これを使用することができる。

【0047】[PnPブローカ機能コンポーネント] PnPブローカの内部コンポーネントを図4に示す。通常、PnPブローカは、サービスレジストリ305、サービスディスカバリ/アベイラビリティエージェント304、サービスセッション管理エージェント306、サービスロケーションプロトコル（SLP: service location protocol）ユーザエージェント308およびSLPサービスエージェント307からなる。PnPブローカ間プロトコル303は、他のPnPブローカ301との通信に使用される。PnPブローカインタフェース302は、ユーザアプリケーション103に対するインタ

フェースを提供する。サービスロケーションプロトコル（SLP）は、当業者に周知であり、James Kempf and Pete St. Pierre, "Service Location Protocol for Enterprise Networks: Implementing and Deploying a Dynamic Service Resource Finder", John Wiley and Sons, 1999, に記載されている。サービスレジストリは、1個以上のサービスレコードを含む。サービスレジストリ内のすべての情報は、検索、索引づけおよび交換を容易にするために、XML（eXtensible Markup Language）として記述される。XMLもまた当業者に周知であり、Natanya Pitts-Moultis and Cheryl Kirk, "XML black book", Coriolis Group Books, 1999, に記載されている。

【0048】[サービスレジストリ] PnPブローカは、サービスについての情報を保持するためにサービスレジストリを含む。最小限、レジストリは、そのPnPブローカに直接に接続されたゲートウェイおよびサービスについての情報を格納する。図3に示すように、これらのサービスは、ローカルPnPブローカ104またはリモートPnPブローカ1104にある。さらに、PnPブローカレジストリは、他のPnPブローカに登録されているサービスについての情報を格納することも可能である。この拡張されたレジストリ機能により、ローカルPnPブローカは、サービスのローカルなディレクトリを保持することが可能となる。これは、ローカルな環境にとって重要であり、ネットワーク内に別個の「集中化した」SLPディレクトリエージェントは不要となる。例えば、ローカルPnPブローカがプリントサーバをサポートする場合、レジストリは、すべての準拠プリントサーバについての情報を有することが可能である。最終的に、この情報は、負荷均衡、フォールトトレランス、あるいはキャッシュのために使用可能である。また、PnPブローカサービスレジストリは、デバイスタイプにかかわらず、ローカルPnPブローカの範囲内にあるすべての準拠デバイスについての情報を提供する、ネットワークディレクトリとして作用することも可能である。この場合、ネットワーク内のPnPブローカのうちの1つが、すべてのPnPブローカを見つけて登録する責任を負う中心ディレクトリとして指定されることになる。ネットワークリソースに対する他のデバイスからのすべての要求は、このPnPブローカへ向けて送られ、このPnPブローカはそれぞれに応答することになる。

【0049】[サービスレコード] サービスレコードは、ネットワーク接続されたクラスタ内の利用可能なサービスユニットのセットであり、利用可能なサービスと、他のPnPブローカから要求されたサービスとを記述する。サービスレコードは、そのローカルクラスタ内の0個以上のサービスユニットレコードからなる。サービスユニットレコードは、サービスユニットIDフィー

ルドと、それに続く0個以上の属性レコードからなる。サービスユニットIDフィールドは、サービスユニットのタイプと、そのサービスユニットによって提供されるサービスとを識別する。デバイスは、ただ1つのサービスユニットを有するが、複数の属性を有することも可能である。属性レコードは、属性IDフィールド、その値、および、アクセス制御情報からなる。属性レコードは、主に、それぞれのサービスまたはデバイスに対する細かいアクセス制御を実施するために使用される。さらに、属性レコードは、デバイスの現在の状態、および、それを変更するために使用可能なインタフェースの記述を格納する。

【0050】[サービスディスカバリ/アベイラビリティエージェント] PnPブローカは、リモートPnPブローカを発見し利用可能なサービスを識別するために、サービスディスカバリ/アベイラビリティエージェントを必要とする。サービスディスカバリ(発見)は、ローカルPnPブローカによって指定される、要求されるサービスタイプを、リモートPnPブローカ上で利用可能なサービスタイプと比較することによって実行される。ローカルPnPブローカからリモートPnPブローカへ要求サービスタイプを送信し、リモートPnPブローカからローカルPnPブローカへその応答を送信するために、上記のJavaRMIやHTTPを使用することができる。要求サービスタイプの指定の検査により、PnPブローカは、リモートPnPブローカに登録されているすべてのあるいは特定のサービスの特性を決定することができる。サービスディスカバリ/アベイラビリティエージェントは、SLPユーザエージェントおよびSLPサービスエージェントの上に位置する。

【0051】さらに、PnPブローカは、サービスディスカバリ/アベイラビリティエージェントを用いて、サービスのアベイラビリティ(利用可能性)を周期的にチェックすることができる。ローカルPnPブローカは、適当なPnPブローカに対して、アベイラビリティチェックを実行するよう要求する。本発明の実施例では、ユーザは、アベイラビリティチェックの周期を指定し、また、このチェックを取り消すことができる。

【0052】[サービスセッション管理エージェント] ユーザアプリケーションがサービスまたはデバイスを発見し、それを使用したいとき、PnPブローカは、ユーザアプリケーションとそのサービスの間に仮想パイプ(トンネル)を確立する。これをサービスセッションという。このようなトンネルを用いて、ユーザアプリケーションとサービスの間で、コマンド、応答およびデータを交換することができる。デバイスのインタフェースの編成に従って、これらのコマンド、応答、およびデータは特定のフォーマットを有し、定義されたプロトコルのもとで交換される。PnPブローカは、この仮想パイプを管理しながら、以下の3つのプロトコルのうちの1つ

で動作するよう指令されることが可能である。

【0053】[ネイティブパケットでのネイティブデータ転送] PnPブローカは、データパイプを設定した後、バックグラウンドに入ることにより、ユーザアプリケーションとサービスが、メッセージストリームおよびデータフォーマットを管理することを可能にする。このアプローチは、PnPブローカが他のネットワークエンティティの能力を発見するために単独で使用され、アプリケーション、サービス、およびデバイスがユーザアプリケーションと発見されたサービスとの間の相互作用を管理するときに有用である。メッセージは、PnPブローカの関与なしに、ユーザアプリケーションとサービスの間で直接に交換される。メッセージ交換は、厳密に、ネイティブパケットにおけるネイティブデータの形式である。サービスを要求する前にサービスを発見するためあるいはサービスの問合せ(クエリ)を行うために、ユーザアプリケーションは、PnPブローカ間プロトコルを使用可能である。例えば、IEEE1394対応のカメラは、このデータ交換フォーマットを用いて、データストリームをデジタルVCRに送ることができる。

【0054】[PnPブローカのパケットにおけるネイティブデータ転送(トンネリング)] データフォーマットはユーザアプリケーションおよびサービスによって選択され制御される一方で、PnPブローカが、データパイプを設定し、メッセージストリームを管理することも可能である。このアプローチは、ユーザアプリケーションと発見されたサービスとの間で共通のメッセージングプロトコルが存在しないときに有用である。すべてのメッセージは、PnPブローカ間プロトコルによって運ばれる。例えば、IEEE1394対応カメラは、PnPブローカを通じて、異なるクラスタ内のデジタルVCRにデータストリームを送る。

【0055】[PnPブローカのパケットにおけるPnPブローカのデータ(完全機能ゲートウェイ)] PnPブローカが、データパイプを設定し、メッセージストリームを管理し、ユーザアプリケーションとサービスとの相互作用のためのデータフォーマット定義を提供する。また、ブローカは、ユーザアプリケーションと発見されたサービスとの間の共通のメッセージングプロトコルおよび共通のデータフォーマットを提供する。メッセージ交換情報は、PnPブローカデータに含まれ、これはパケットにフォーマットされる。ユーザアプリケーションメッセージは、PnPブローカを通るが、PnPブローカは、メッセージの内容あるいはセマンティクスを検査することはない。このタイプの相互作用の例は、USBデバイスと対話するIEEE1394対応デバイスである。

【0056】[SLPユーザエージェント] ユーザエージェントは、あるサービスとの接続を確立するためにユーザに代わって動作するプロセスである。ユーザエー

エージェントは、サービスエージェントまたはディレクトリエージェントからサービス情報を取得する。

【0057】[SLPサービスエージェント] サービスエージェントは、サービスを公表するために、1個以上のサービスに代わって動作するプロセスである。

【0058】[プロトコルと相互作用] PnPブローカは、ユーザアプリケーションがネットワークのプラグアンドプレイ機能を利用することができるようにする標準化されたインタフェースを公開する。このPnPブローカインタフェースを用いて、アプリケーションは、トランスポート層とは独立にPnPブローカ間プロトコルを用いて相互に通信するように書くことができる。この設計は、ポータビリティ（可搬性）を促進し、ネットワークサービスの発見および選択のためのスケーラブルなフレームワークを提供する。PnPブローカ間プロトコルは、IETF Service Location Protocol v.2に基づいており、サービスをサポートするネットワークホストの名前をユーザアプリケーションが知ることは不要である。むしろ、ユーザは、PnPブローカインタフェースを通じて、所望のサービスタイプと、対応する属性のセットとをPnPブローカに与える。これらは、PnPブローカに対して、所望のサービスを記述する。この記述に基づいて、PnPブローカは、PnPブローカ間プロトコルを用いてサービスのネットワークアドレスを解決し、PnPブローカ間プロトコルは、サービスロケーションプロトコルを使用する。また、PnPブローカインタフェースは、デバイスあるいはサービスのためのユーザの識別および認証機構を扱う。いったんサービスが識別されると、PnPブローカ間プロトコルは、Java RMIあるいはHTTPを用いて、ユーザアプリケーションが、発見されたサービスを使用することを可能にする。このフレームワークのもう1つの重要な特徴はゲートウェイによって提供される。ゲートウェイは、デバイスの能力をPnPブローカに登録し、PnPブローカが、PnPブローカにおいてサービスエージェントを通じてサービスを使用することを可能にする。

【0059】要約すれば、本発明のプロトコルスイートは、以下のプロトコルおよびインタフェースからなる。

- ・PnPブローカとユーザアプリケーションの間のインタフェース。
- ・あるPnPブローカと別のPnPブローカの間のプロトコル。
- ・PnPブローカとゲートウェイの間のプロトコル。
- ・ゲートウェイとダミーの間のプロトコル。

【0060】[ユーザアプリケーションへのPnPブローカのインタフェース (PnPブローカインタフェース)] PnPブローカは、サービス登録、サービス発見、サービス要求、およびアベイラビリティチェックのために、ユーザアプリケーションに対して、標準化されたAPIおよびインタフェースを公開する。さらに、P

nPブローカは、適当なサービスあるいはデバイスのためのユーザの識別および認証を扱う。

【0061】[サービス登録] 必要なとき、サービスユニットまたはユーザアプリケーションは、それぞれ、サービスユニットまたはユーザアプリケーションとしてローカルPnPブローカに自分を登録または登録解除するために、RegisterService()またはUnregisterService()を呼び出す。ユーザアプリケーションまたはサービスユニットがPnPブローカに登録された後、その機能は、別のユーザアプリケーションまたはサービスユニットからのQueryService()またはSearchService()要求に対する応答に含まれる。

【0062】[サービス発見] ユーザアプリケーションは、SearchService()を呼び出して、ローカルPnPブローカに対し、特定のサービスを有する登録されたサービスユニットを含むPnPブローカを探索するよう要求する。ユーザアプリケーションは、QueryService()を呼び出して、どのようなサービスユニットが登録されているか、および、ユーザアプリケーションによって指定されたPnPブローカに登録されているサービスユニットの機能を発見する。

【0063】[サービス要求] ユーザアプリケーションまたはサービスユニットは、OpenService()を呼び出して、ローカルPnPブローカに対し、ローカルPnPブローカまたはリモートPnPブローカのいずれかに登録されている特定のサービスユニットとのサービスセッションを開始するよう要求する。TransferData()、ReceiveData()、およびCloseService()コールにより、追加機能が提供される。

【0064】[アベイラビリティチェック] ユーザアプリケーションまたはサービスユニットは、StartAvailabilityCheck()を呼び出して、ローカルPnPブローカに対し、ローカルPnPブローカまたはリモートPnPブローカのいずれかに登録されている特定のサービスユニットが動作しているかどうかを周期的にチェックするよう要求する。

【0065】[ユーザの識別および認証] 好ましい実施例では、本発明は、Javaアプリケーション環境を利用する。これは、分散コンピューティングのための適切な、現在利用可能なコンピューティングプラットフォームを提供するからである。この環境では、コードおよびデータの両方が、マシン間で移動可能である。この環境は、別のマシンからダウンロードされたコードが、妥当なレベルの信頼性で動作することを可能にする組み込みセキュリティを有する。Javaアプリケーション環境における強い型付けにより、仮想マシン上で、オブジェクトのクラスの識別は、そのオブジェクトがそのマシン上で発生したものでないときでも実行可能である。その結果、ネットワークは、必要に応じて移動可能なオブジェクトの流動的なコンフィグレーションをサポートし、オ

ペレーションを実行するためにネットワークの任意の部分呼び出すことができる。

【0066】ホームデバイスへの認証なしのアクセスは、フレームワーク全体のセキュリティにとって重大な結果を生じる可能性がある。リソースへの制御されたアクセスは、安全性とセキュリティの基礎である。従って、本発明の一実施例は、Java仮想マシンおよびJavaプログラミング環境を使用して、1つのデバイスのセキュリティを保証する。システムは、障害のあるデバイスに対処するように設計することが可能であるが、ネットワークセキュリティは、デバイスセキュリティよりも複雑な問題となる。

【0067】本発明のもう1つの実施例は、コードを参照する手段として、および、基本的なアクセス制御機構として、コードのMD5チェックサムを使用する。MD5は、コードのチェックサムを生成する暗号化手続きであり、アクセス制御のために使用される。しかし、ネットワークが大規模になると、セキュリティポリシーを指定しコードに名前付けするためのより柔軟な分散機構が必要である。

【0068】本発明のセキュリティモデルは、プリンシパルとアクセス制御リストという2つの概念の上に構築される。サービスは、あるエンティティ（プリンシパル）に代わってアクセスされる。プリンシパルは、一般に、システムの特定のユーザに由来する。サービス自体が、そのサービスを実装するオブジェクトの識別に基づいて、他のサービスへのアクセスを要求することが可能である。サービスへのアクセスが許可されるか否かは、そのオブジェクトに対応するアクセス制御リストに依存する。

【0069】[PnPブローカ間通信プロトコル] PnPブローカ間通信プロトコルは、2つの部分に分かれる。第1に、ネットワーク内で新たなデバイスおよびサービスを発見するための、プロトコルおよび機構のセットが使用される。発見（ディスカバリ）手続きは、ユーザアプリケーションがどのようにしてネットワークインフラストラクチャを探索するか、および、ユーザアプリケーションがどのようにして自分自身およびそのサービスを登録することができるかを指定するサービスディスカバリプロトコルの部分に基づく。他方、ルックアップ手続きは、ユーザアプリケーションが、集中化されたレジストリがある場合あるいはない場合に、必要とするサービスを探索するためにどのようにしてレジストリに問合せを行うかを指定するプロトコルに基づく。

【0070】いったんサービスが探索された後、そのサービスを利用するために他のステップが必要になることがある。本発明の実施例では、ユーザアプリケーションは、サービスの品質およびセキュリティ条件を含めて、サービスへのアクセスの交渉を行う。さらに、サービスへのアクセスの交渉が成功した後、ユーザアプリケーシ

ョンは、Java RMI、HOP、あるいはHTTPプロトコルを使用して、実際にサービスを利用する。

【0071】[サービスディスカバリ/登録プロセス] ルックアップおよびディスカバリの両方のために、サービスロケーションプロトコル（SLP）が使用される。図5に、サービスディスカバリ手続きを示す。SLPは、自動的に、しかも事前の設定なしで、ユーザエージェント（UA：user agent）502の要求を、サービスエージェント（SA：Service Agent）504によって提供されるサービス512と照合する。SLPは、SAによるサービスの公表と、ディレクトリエージェント（DA：Directory Agent）511により管理されるSLPディレクトリ508へのサービスの編成とを処理する。また、SLPは、サービスがユーザアプリケーションにサービスの能力および設定条件を通知する手段を提供する。

【0072】PnPブローカ501内のUA502は、ユーザアプリケーション103によるサービスおよびリソースの要求を理解する。同じくPnPブローカ503（同じPnPブローカでも異なるPnPブローカでもよい）にあるSA504は、各ネットワークサービスを代表し、UA502にとって利用可能なサービスを行う。SLPは、動的にサービス属性を保持するため、UA502は、現在の情報を取得することが可能である。サービスは、アプリケーションによって自動的に探索されることも可能であり、あるいは、サービスブラウザでユーザに提示されることも可能である。SLPは、既存のアプリケーションをサポートするとともに、新たなサービスの公表および発見を容易に可能にする。

【0073】本発明によれば、サービスは、ゲートウェイによって記述される。ゲートウェイは、SA内の属性（そのサービスに対応する）の値を設定する。例えば、プリンタは、ポストスクリプトプリンタとして、青色の紙が利用可能なプリンタとして、あるいは、ユーザのオフィスの同じフロアにあるプリンタとして、記述することができる。UA502は、DA511への要求メッセージSrvReq505において必要なキーワードおよび属性値を指定することによって適当なプリンタを選択し、応答を待つ。DA511は、適当なサービスのネットワークロケーションを含む応答SrvRply505により応答する。小規模な施設では、DAがないことも可能であり、その場合、要求メッセージは直接にSAに送られることになる。これを、ブロードキャストによるディスカバリ（507）という。

【0074】サービスは、DA511に登録することによって自分自身を公表する。サービスの登録は、サービスを記述するすべてのキーワードと属性値対（属性と値からなる対）とからなるリストを含む。また、登録は、リソースへのリースを含む。これにより、リースの満期後、サービス情報はDA511によって削除される。リ

ース機構は、サービスが永久に公表され続けているのにサービスハードウェアがもはや利用可能でない状況为了避免のために実装される。また、明示的な登録解除も、SAの規則正しいシャットダウン手続きの一部として、DAからサービスの情報を削除することが可能である。また、SAが、現在の属性情報を周期的に登録することにより、UAが、サービスに関連するステータス、負荷、温度、あるいはその他の動的特性を確認することも可能である。

【0075】本発明のフレームワークでは、サービスは、そのサービスタイプに従って分類される。各サービスタイプは、SA504がサービスを記述するために利用可能にする、利用可能なキーワードおよび属性のセットを定義する。サービスブラウザは、まず、UA502にとって利用可能なサービスタイプを発見した後、そのサービスタイプについてSAによって公表されているすべての情報を問い合わせることによって、UA502にとって利用可能なすべてのサービスのすべての特性を決定することができる。

【0076】[ユーザエージェントとサービスエージェントの相互作用] サービス/デバイス登録プロセスの一部として、まず、サービスユニット512のゲートウェイは、サービスあるいはデバイスをSA504に登録し、その後、SA504は、SrvRegメッセージ506をDA511に送る。この登録は、サービスの特定のインスタンスに対するサービスURIと、そのサービスを記述する属性値対とを含む。DA511は、さまざまな目的で、このような登録をキャッシュすることが可能である。登録がキャッシュされた後、DA511は、SrvAckメッセージにより応答する。また、サービス登録は、寿命(lifetime)を含む。サービスが利用不能になったがそれ自身を登録解除することができなかった場合、寿命値により、DA511は、キャッシュされた登録を満期にすることができる。この状況は、SA504がSrvDeregメッセージ506を発行することができない場合にしか生じないはずである。正常動作中、SA504は、後続のSrvRegメッセージで周期的にサービスの登録をリフレッシュする。このような「リフレッシュ」メッセージは、いずれかの値が変化した場合には更新された情報を含むことが可能であるが、完全な属性のセットを含む必要はない。

【0077】PnPブローカ501のUA502は、サービスが要求されるときに、DA511に対して要求を行う。UA502がDA511を発見することが可能ないくつかの方法がある。起動時にDA511のアドレスをUA502に静的に設定することに加えて、UA502は、当業者に周知のDHCP(Dynamic Host Configuration Protocol)を用いてDA511のアドレスを要求することも可能である。第3のオプションとして、リンク上にマルチキャストを行い、DAの公表を受け取る

いうことがある。これらの3つのオプションは共存するが、SLPが、手動設定なしで動作可能であることが重要である。この理由で、UA502は、DA511を見つけるためにSrvReqを使用することが可能である。SrvReqは、SLPに対するIANA割当てマルチキャストアドレスにマルチキャストされる。IANAは、当業者に周知のInternet Assigned Numbering Authority標準である。これはマルチキャスト要求であるため、複数のユニキャスト応答を受け取る可能性がある。その結果として得られるDAのリストとは、他のサービス要求のために使用可能である。

【0078】いったんUA502がDA511のアドレスを得ると、後続のサービス要求は直接にそのエンティティに対して行うことが可能である。プリンタを例とすると、UA502がプリンタサービスを探索しようとする場合、SrvReqが作成される。このメッセージは、サービスタイプ“printer”と、オプションの属性および値のリストとを含む。属性値対は、要求されるプリンタのタイプを記述する。このメッセージは、事前に設定された、または、マルチキャストを通じて発見された、DA511へユニキャストされる。DA511は、SrvReqを受信し、キャッシュされた登録に対するルックアップを実行して、要求された属性および値と一致するものを見つけようと試みる。その後、SrvReplyが、要求側のUA502へユニキャストされる。この応答は、一致結果に依存して、0個以上のサービスURIを含む。その後、ユーザアプリケーションは、そのサービスURIを用いてプリンタを見つける。

【0079】施設内にDA511がない場合、PnPブローカは、同じくラスタ内のSA504を見つけることに制限される。これは、多くの小規模な施設では許容できるかもしれないが、PnPブローカのスケラブルなアプリケーションの場合、および、複数のクラスタを有する大規模な施設では、DA511を使用しなければならない。さらに、SLP DA511は、LDAPベースのディレクトリ509へのゲートウェイであることが可能であるため、PnPブローカインタフェースは、変換スキーマ510を用いて、すべてのプロトコルに対する単一のAPIを提供する。

【0080】UA502は、サービスハンドルを取得したいとき、サービスタイプと、要求する属性の文字列ベースの間合せとをサービス要求として送信する。サービス応答が返されるとき、これは、サービスを指すURIを含む。URIは、SA504のIPアドレスを含むか、さもなければ、DNSがIPアドレスへと解決することができる名前を含む。また、URIは、UA502がサービスを使用するために必要とするその他の情報を含む。例えばプリンタの場合、キュー名と、プリントサービスをホスティングしているコンピュータのアドレスとが返される。URIは、リソースロケーションを表す

ための従来の標準と同様の自然な方法で、サービスを見つけるために使用される。さらに、文字セット問題は、標準的な方法で対処することができる。

【0081】[PnPブローカとユーザエージェントおよびサービスエージェントとの相互作用] SLPを使用することにより、PnPブローカは、同じタイプの多くの他の公表されたデバイスから適当なサービスの正確な選択をすることができる。これは、UA502によって要求されたキーワードおよび要求された属性値に一致するサービスのみを要求することによって行われる。このようなキーワードおよび属性値は、AND、OR演算子、一般的な比較演算子により、また、部分文字列マッチングを使用することにより、ブール式へと結合することができる。

【0082】PnPブローカは、ユーザへのコンフィグレーション情報の提供を透過的にし、新たなサービスの設定の作業を容易にする。これらはいずれも、システムアドミニストレータを必要とする。SA504が近隣のDA511にサービスを公表するように設定された後、UA502は、さまざまなサービスがオペレーションを開始および停止するとともにネットワーク上で変化する条件に、動的に適應することができる。例えば、ウェブアプリケーションは、現時点でたまたま動作中であるシステムとは独立に、適当なプリンタが利用可能である限りそれを見つけることができる。

【0083】ゲートウェイは、SA504に新たなデバイスあるいはサービスを追加したいとき、そのサービスの利用可能な属性およびキーワードを供給する。SA504は、プログラマ的にSLPに登録を行い、サービスの固有のコンフィグレーション情報に基づいて属性に対する値を割り当てることができる。その後、SLPは、登録(公表)を処理し、ユーザアプリケーションとサービスの間のコネクションの確立を可能にする。

【0084】新たなサービスタイプを必要に応じて定義することができる。実験的なサービスタイプを限定された用途で配備し、その後一般用に公開することも可能である。これは、SLPのネーミング権限機能を用いて実行される。ほとんどの一般的なサービスタイプは、デフォルトで、IANA(Internet Assigned Numbering Authority)により標準化されたサービステンプレート(スキーム)定義を使用する。代替サービスタイプを定義することは、任意の代替ネーミング権限に知られているスキーム定義(通常は、ローカル管理者に知られている名前)を使用するようにディレクトリエージェントを設定するのと同様に容易である。例えば、家庭でセキュリティサービスを提供するためには、"service:secure:Door"というサービスタイプを定義することができる。もちろん、このタイプのサービスハンドルを利用するユーザアプリケーションは、"secure"サービスと通信するために、そのサービスのネットワークプロトコルを使用しな

ければならず、さらに、この特別のサービスタイプのURIに含まれる情報を解析することができなければならない。これは、任意のユーザアプリケーション/サーバプロトコルの自然な動作に本来的なことである。

【0085】[SLPをLDAPとともに使用すること] LDAPv3(Lightweight Directory Access Protocol)は、当業者に周知であり、汎用のディレクトリアクセスプロトコルとして普及しつつある。LDAPはその名前にLightweight(軽量)とあるが、SLPはLDAPよりもさらに「軽量」である。また、SLPは、自動ディレクトリ管理を提供するため、および、階層的で自由度の少ない名前空間における不適切なリソース配置を必要としないために、リソース管理においてLDAPよりすぐれている。SLPにおいて新たなサービスタイプを追加することはLDAPの場合よりもずっと容易である。タイプによる問合せは、SLPの場合のほうがLDAPの場合よりも効率的である。既存のリソースに対する属性記述は、SLPでは利用可能であるが、LDAPでは不可能である。

【0086】SLPでは、事前コンフィグレーションなしでのディスカバリが可能である。これに対して、LDAPは、はじめに、ディレクトリのアドレスと、LDAPが使用するディレクトリスキームの知識を必要とする。SLPでは、非標準的な属性と、標準化されたサービスタイプテンプレートの新しいバージョンが可能であるため、進化することができる。

【0087】本発明の実施例は、SLPを使用して、LDAPディレクトリの管理を簡単化し、SLPが、必要に応じてLDAPディレクトリから取得される情報を返すことを可能にする。別の実施例では、SLPは、LDAPにサービスエントリを入力するために動的に使用され、SLP問合せはLDAP問合せにマッピングされて、LDAPがSLPのバックエンドとなる。このようなコンフィグレーションの1つの利点は、ユーザアプリケーションが、LDAPディレクトリから直接にユーザ認証情報を得ることである。

【0088】[サービス利用プロセス] サービスディスカバリ/登録プロセスは、サービスローケーションプロトコルによって提供されるURIによって検索可能なサービスレコードを提供する。PnPブローカ間の交換能力は、サービスレコードを含むQueryService()メッセージを交換することにより提供される。この問合せ(query)は、ユーザアプリケーションが利用したいサービスの要件を記述し、リモートPnPブローカに対してサービスの詳細を要求する。

【0089】受信側PnPブローカは、QueryService()メッセージを受信すると、受信したサービスレコードを自己のサービスレコードと比較する。その後、受信側PnPブローカは、一致したサービスレコードを含むサービスレコードを要求側PnPブローカに返す。このアプ

ローチは、すべてのサービスユニット、特定のタイプのサービスユニット、あるいは、特定の能力のセットを有するサービスユニットのうちで、ユーザアプリケーションが探索を行う技術を提供する。ユーザアプリケーションは、サービスレコードを受け取った後、サービスあるいはデバイスを使用するためにPnPブローカとの通信を開始することができる。

【0090】ローカルPnPブローカは、リモートPnPブローカへメッセージを送ることによって、ユーザアプリケーションとサービスの間のサービスセッションの確立を要求する。このメッセージは、ユーザアプリケーションサービスハンドル、リモートサービスユニットハンドル、プロトコルID（ネイティブ/トンネル/ゲートウェイ経由）、要求側（リクエスト）ID、および要求側資格証明(credential)からなる。

【0091】リモートPnPブローカは、要求されたサービスが受容されるかそれとも拒否されるかを指定するリザルトコードで応答する。リモートPnPブローカは、ある時間後のリダイレクトまたはリトライを含むことも可能である。サービスが受容される場合、リモートPnPブローカは、サービスセッションの開始を識別するサービスハンドルを送信する。サービスセッションは、ユーザアプリケーションとサービスユニットの間、または、2つのサービスユニットの間でのデータ転送を扱う。ユーザアプリケーションは、サービスの使用を完了すると、ローカルPnPブローカに対して、“close service”（サービス終了）メッセージをリモートPnPブローカへ送るよう要求する。さらに、ユーザアプリケーションは、サービスが利用可能であるかどうかを周期的に判定する必要があるとき、それぞれのPnPブローカに対して、フレームワーク内のPnPブローカ間でアベイラビリティチェックを実行するよう要求することができる。アベイラビリティチェックは、1つのデバイス全体に対しても、あるいは、デバイス内で提供される特定のサービスに対しても、実行可能である。

【0092】[PnPブローカ/ゲートウェイプロトコル] PnPブローカ/ゲートウェイプロトコルは、簡単なメッセージ交換機構に基づいている。ゲートウェイは、既に説明したように、ゲートウェイに直接に接続されたサービスユニットを代表し、提供されるサービスを登録する。また、ゲートウェイは、PnPブローカにあるサービスエージェントに対する、任意のサービスのコンフィグレーションにおいてなされる変更を反映する。この登録は、リースに基づいており、ゲートウェイまたはサービスエージェントのいずれかにより更新可能である。登録は、サービスエージェントに現在の情報を提供する。PnPブローカは、特定のサービスに対する要求を受け取ると、まず、サービスエージェントをチェックして、必要なデバイスあるいはサービスをいずれかの直接接続されたゲートウェイが提供するかどうかを判定す

る。また、ゲートウェイにより、サービスエージェントは、ゲートウェイ内の特定のイベントに関心があることを登録し、そのイベントの発生の通知を受け取ることが可能である。このようなイベントは、ネットワークにおけるデバイス/サービスの追加/削除であることが可能である。サービスのサービスレコードがアプリケーションの要件に一致した場合、ユーザアプリケーションは、ゲートウェイ内のサービスユニットと、ユーザアプリケーションとの間のサービスセッションを設定することができる。

【0093】[イベント通知機構] 本発明のフレームワークでは、ユーザアプリケーションは、イベント通知要求をサーバに登録することができるため、サーバは、新たなデバイスがオンラインになったとき、あるいは、デバイスの属性が変化したときに、ユーザアプリケーションに通知することになる。これを達成するため、本発明は、登録ユーザにイベント通知を配信するための機構を提供する。

【0094】ネットワークイベント通知プロトコルには、CORBA (Common Request Broker Architecture) イベントサービス、X-Windowシステムイベント、SGAP、BSCWなどのいくつかの周知のものがある。例えば、CORBAイベントサービスは、Robert Orfali and Dan Harkey, “Client/Server Programming with Java and CORBA”, Wiley, 1996、に詳細に記載されている。

【0095】しかし、上記のイベント通知サービスは、特定のアーキテクチャで動作するように設計され、大規模なコードベースを課するため、軽量通知サービスにおいて実際に使用するのは困難である。

【0096】いくつかのタイプのイベントの通知のための登録に関して提供される能力に加えて、ユーザは、属性を通知要求に関連づけることができる。好ましくは、このような通知は、高い信頼性で配信され、完全に規則正しいエンドツーエンド配信を保証すべきである。しかし、通知が、信頼できないトランスポートを通じて配信されるとき、確認応答やリトライは不要である。ユーザアプリケーションは、イベントの需要側が、受信の明示的な確認を供給側に提供することを要求することも可能である。認証のために、ユーザは、通知にデジタル証明して、通知の正当性および完全性を保証することができる。

【0097】本発明のフレームワークは、SLPの使用により、2つの方法でイベント通知を達成する。第1実施例によれば、サービスエージェントは、サービスが更新された後、マルチキャストアナウンス (Subscribe) を送信する。更新を受信した後、ユーザエージェントは、ブラウザ更新を行うことが可能であり、DAは登録を更新することができる。このような使用法は、SLP使用では明示的に禁じられていないが、フラッディング

グ(flooding)を引き起こす可能性がある。さらに、このアプローチは、多数のサービス、異なる言語、およびデジタル署名に関して効率的に動作しない。また、ServRegは、ネットワークにおいて追加/削除/変更された各サービスのURIおよび属性を含む。

【0098】本発明のもう1つの実施例では、サービスエージェントは、ネットワーク内のすべてのユーザエージェントへのブロードキャストにより、SAAadvertisesメッセージをアナウンスする。SAAadvertisesメッセージは、提供されるすべてのサービスタイプのリストを含む。SAAadvertisesを得た後、ユーザエージェントは、送信側サービスエージェントへ適当なサービス要求をユニキャストする。サービスエージェントからのアナウンスは、指数バックオフ方式で行われる。

【0099】[ゲートウェイ/ダミープロトコル] ゲートウェイ/ダミープロトコルは、特殊なプロトコルであり、ネットワークにより統合されるデバイスのタイプを反映する。ネットワークは、X-10、USBおよびIEEE1394のデバイス/サービスをそれぞれネットワークの残りの部分に接続するX-10ゲートウェイ、USBゲートウェイ、およびIEEE1394ゲートウェイからなることが可能である。ダミーデバイスは、サービスレコードに対応するタイプのレコードを独自(proprietary)フォーマットで格納する。ゲートウェイ/ダミープロトコルは独自のものであるが、PnPブローカに対しては、標準化されたインタフェースを公開する。

【0100】[実施例] 本発明の一実施例によれば、ネットワークは、1個以上のPnPブローカを有する。PnPブローカは、ユーザといくつかのサービスとの間のインタフェースとして作用する。一実施例では、X-10デバイスに対するPnPブローカは、USBデバイスによって提供されるサービスにリンクされたいくつかのボタンあるいはその他のユーザインタフェースウィジェットを提供する。ここで用いた「ユーザインタフェースウィジェット」という用語の意味には、ユーザアプリケーションのグラフィカルユーザインタフェースを作成するために使用されるボタン、ダイアログ、テキストウィンドウ、スケールおよびその他のグラフィカルコンポーネントを含む。あるボタンがクリックされると、PnPブローカは、単に、X-10ゲートウェイを通じて、特定のサービスを呼び出す。以下のステップは、この実施例により、ユーザアプリケーションが、ネットワーク内の新たに登録されたサービスを利用するために必要とされる。

【0101】1. 白熱電球が、家庭の電源アウトレットに挿入される。

【0102】2. 白熱電球は、ダミーデバイスであるため、X-10デバイスを通じて電源線ネットワークに接続される。

【0103】3. ホームネットワークでは、PnPブロー

ーカおよびX-10ゲートウェイは、新たなデバイスの導入前に既に動作中である。

【0104】4. X-10ゲートウェイは、利用可能なアドレスを周期的にポーリングし、いずれかのデバイスがアクティブになったかどうかをチェックする。

【0105】5. ゲートウェイは、新たなデバイスがネットワークに入ったことを認識し、そのデバイスと、そのデバイスにより提供されるサービスとの両方を、PnPブローカのサービスエージェントに登録する。

【0106】6. サービスエージェントは、デバイスを使用するために、あるリースを取得する。このリースは、各リース期間後に更新して、これにより、ゲートウェイに対して、サービスが依然として利用可能であるかどうかをチェックするよう要求しなければならない。

【0107】7. サービスエージェントは、デバイスにより提供されるサービスを、サービスロケーションプロトコルを用いて、ディレクトリエージェントに登録する。

【0108】8. ユーザは、デバイスを使用したい場合、PnPブローカインタフェースを用いて、要求するサービスに対する問合せおよび検索を行う。すると、PnPブローカにあるユーザエージェントは、ディレクトリエージェントと連絡をとるか、あるいは、(サービスエージェントおよびユーザエージェントがいずれも1つのPnPブローカ内にある場合には) 直接にサービスエージェントへ行く。

【0109】9. 新たに発見されたデバイスが、ユーザのデバイス記述と一致する場合、サービスエージェントあるいはディレクトリエージェントは、サービスパラメータとともに、固有のURIをユーザエージェントに返す。

【0110】10. ユーザアプリケーションは、これが非TCP/IPデバイスであると判定し、これに従い、PnPブローカ内のサービスセッション管理エージェントは、ユーザアプリケーションとデバイスの間のセッションを確立する。しかし、このセッションでは、コマンド/データ転送は、PnPブローカをゲートウェイとして使用して、PnPブローカ間プロトコルにより行われる。

【0111】11. また、ユーザアプリケーションにより提供されるインタフェースは、デバイス内の機能の実行に関する情報も含む。

【0112】12. ゲートウェイは、サービスにおける状態の変化をサービスエージェントに知らせて更新する。

【0113】図6において、本発明の一実施例による例示的なホームネットワークは、電源線(X-10)クラスタ613、電話線(HomePNA/xDSL)クラスタ602および606、CAT5(イーサネット(登録商標))クラスタ616、USBクラスタ610、な

らびにJ i n iデバイスクラスタ609などの、周知のネットワークコンポーネントにより構成される。レジデンシャル（住宅）ゲートウェイ605は、イーサネット、Home PNA、xDSL、およびJ i n iを含むさまざまなネットワークを相互接続するために使用される。イーサネットクラスタは、ユーザのホームPC620を含む。ホームPC620は、イーサネットプロトコルにより、デジタルカメラ617およびウェブパネル618のようなさまざまなデバイスに接続される。ウェブパネル618は、NEC社の製品であり、タッチパッドを有し、ユーザに、インターネットに容易にアクセスする手段を提供するデバイスである。イーサネットクラスタ616は、イーサネットハブ619を通じてレジデンシャルゲートウェイ605と通信する。イーサネットハブ619もまた、インターネットへのホームネットワークのブリッジを提供する。また、ホームPC620は、USB配線を用いてスピーカ611およびネットワークカメラ612に、および、X-10配線を用いて白熱電球614およびセキュリティマネージャ615に接続される。さらに、ホームネットワークは、別のホームPC603および607、テレビジョン601、ならびに電話機604を有する。ホームPC607を含むxDSLクラスタ606は、ブリッジ608を通じてレジデンシャルゲートウェイ605に接続される。PC620上で動作するPnPブローカは、ユーザが、ウェブブラウザに基づくインタフェース内で、新たなデバイス/サービスを検索することを可能にする。本実施例は、ユーザインタフェースとして、Microsoft Internet ExplorerTM v5.0ウェブブラウザを使用するが、他の適当なブラウザも使用可能である。

【0114】デバイス/サービスは、対応するネットワークに挿入されると、設定可能なプロパティとともにブラウザに現れる。デバイス固有のゲートウェイは、デバイス/サービス記述を格納する。ディレクトリエージェントがネットワーク内にある場合、ディレクトリエージェントが、デバイス/サービスの機能の登録を扱う。正当な権限が与えられた後、ユーザは、デバイスのプロパティの変更や、デバイスの相互接続が可能である。PnPブローカは、デバイスに対するユーザの認証と、非互換の場合のデータ/メッセージの変換を扱う。例えば、X-10デバイス614および615は、PC620上にあるゲートウェイデバイスを使用することによって、フレームワーク内の残りのデバイスと対話することができるダミーデバイスである。PnPブローカは、HTTPサーバへのアドオンとして実装されることも可能である。ユーザは、電源線デバイス614および615の状態（電灯のオン/オフ）を変更することができる。デバイス（例えば、カメラ617）がTCP/IPベースである場合、ユーザは、カメラにより提供されるインタフェースに基づいて手動設定を実行し、カメラにより撮ら

れる画像を見ることができる。ユーザは、ビデオオンデマンドサーバに対して、利用可能な映画のリストを問い合わせることができる。通常、（PC603上のMPEG-2デコーダを用いて）ネットワークに接続されたアナログTV601は、同じタイプのインタフェースを使用している場合には、ビデオオンデマンドサーバにも接続可能である。しかし、TV601およびネットワークカメラ612はインタフェースを共有していないため、ユーザは、カメラの画像を直接にTVで見ることはできない。J i n iデバイス（例えば、プリンタ）がネットワークに接続されている場合、このデバイスは、非J i n iデバイスがあたかもJ i n iデバイスであるかのようにして、非J i n iデバイスとともに動作する。

【0115】[J i n iインタフェースの問題点に対する解決法] ネットワークプラグアンドプレイを達成する候補のうちの1つとしてのJ i n iインタフェースモデルの欠点については既に説明した。本発明は、J i n iセキュリティモデルを借りながら、上記のJ i n iインタフェースの欠点を克服する。J i n iのインタフェースの問題点を解決するには、以下の4つのアプローチを使用可能である。第1に、標準に基づく相互運用性が提供可能である。このアプローチによれば、すべてのサービスは標準APIを有し、サービスは、それらの標準APIを実装することになる。第2のオプションは、サンドボックスに基づく相互運用性である。この場合、適当なセキュリティモデルを有するJava仮想マシンが与えられれば、サービスは独立して動作することができる。第3のオプションは、リフレクションに基づく相互運用性である。この場合、アプリケーションは、サービスに対して、そのインタフェースについて質問し、リフレクション機構を通じて、このインタフェースの状態に影響を及ぼす。最後に、第4の方法は、実装に基づく相互運用性である。この場合、同じ人あるいは会社が、プロキシおよびサービスの両方を製造することになる。

【0116】J i n iとは異なり、コードをクライアントに移動しそれをJava仮想環境内で実行する代わりに、本発明は、サービスのアドレスおよびそのサービスのインタフェースを記述するその他のパラメータを処理し、このインタフェース情報をクライアントに提供する。

【0117】本発明のフレームワークが提供する主な利点の1つは、利用可能な機能およびユーザ相互作用モードにおける変更に応じたエンドユーザアプリケーションの動的な適応および再構成である。一部のインタフェースモデルは、ユーザが「従来型」ソフトウェアとのみ相互作用することができることに基づいている。このようなシステムでは、ユーザは、アプリケーションをカスタマイズする能力において制約があるため、アプリケーションは、デバイスの能力を最大限に、すなわち、サービス設計者が予測したユーザの需要あるいは提供したプロ

グラムインタフェースの限界まで、利用することができない。

【0118】多くの場合、自由度はユーザインタフェースによって制限される。アプリケーションの状態およびアプリケーションの初期設定は、ユーザインタフェースを通じてのみ操作可能であるが、ユーザインタフェース自体はあまり設定の自由度はない。これは、モノリシックなプログラムにおいて顕著であり、アプリケーションの状態にフックを設けたり、適切に定義された外部プロトコルを通じてアプリケーションの状態にアクセス可能にしたりするのがきわめて不便である。これに対して、クライアント/サーバプログラムは、公開された相互作用プロトコルを提供するが、2つの大きな問題点を有する。インタフェースの仕様がしばしば臨時的(ad hoc)であること、および、ネットワーク機能についてのアプリケーションのビューを(クライアントコードを修正することによって)変更することができるのはプログラマのみであることである。

【0119】上記のインタフェースの問題点を解決する1つのアプローチは、アプリケーションプログラムが、オンザフライでクライアントデバイスあるいはコンピュータに、例えばJavaアプレットとして、ダウンロードされるようにすることである。しかし、このアプローチの問題点として、エンドユーザが、関連するエンティティとして異種のサービスのセットとの相互作用のためにアプリケーションをカスタマイズすることができないことがある。すなわち、このアプローチでは、アプリケーションが透過的でないために、機能的に同一のサービスの場合でも、プロトコルにおける小さい相違を克服することができない。例えば、上記のインタフェースモデルに基づく電灯スイッチ制御プログラムは、新たな電灯スイッチに遭遇するたびに、それを使用するためには異なるアプリケーションをダウンロードする必要がある。したがって、このアプローチは機能を公開するが、操作しやすい形式ではない。

【0120】もう1つのアプローチは、さまざまなサービスの機能インタフェースを標準化し、アプリケーションがこれらのインタフェース標準をサポートすることを要求することにより、上記の問題点を回避することである。このアプローチの問題点は、多数のアプリケーション固有の標準を強制することが非実際的である点である。

【0121】明らかに、上記のいずれとも異なるモデル、すなわち、インタフェースを公開する要求と、プロトコル標準について合意する要求とのバランスのとれたモデルが好ましい。この問題点を解決するための第1の困難は、サービスの利用可能な機能(インタフェース)を記述する単一ドキュメントスキーマを定義し、関連するプログラムおよびユーザインタフェースをサービスの集合に(およびその逆)関連づけることである。第2の

困難は、カスタムユーザインタフェースが利用可能でないときに、上記のスキーマで書かれたドキュメントを用いてユーザインタフェースを生成し、ランタイム環境を実装することができるソフトウェアを提供することである。

【0122】本発明の一実施例は、コンポーネントベースのアプリケーションフレームワークを、コンポーネント記述のためのアーキテクチャ独立なドキュメントモデルとともに利用する。このようなコンポーネントベースのアプリケーションフレームワークの詳細な説明は、David Krieger and Richard Adler, "The emergence of distributed component platforms", IEEE Computer Magazine, p. 43-53, March 1998, にある。このフレームワークは、上記の2つの基本的なアプローチの特徴を組み合わせることにより、コードフラグメントのアップロード/ダウンロードを可能にし、アプリケーション固有でないインタフェースの記述および操作のために標準を課する。

【0123】[本発明のアクティブコンフィグレーションモデル] 本発明のアクティブコンフィグレーションモデルでは、XML(eXtensible Markup Language)記述が、すべてのネットワークデバイスに関連づけられる。XMLが使用されるのは、XML記述が、(サーバ側での)デバイスの機能の公表として、静的で不変のインタフェース記述を提供するからである。さらに、このようなXMLサービスインタフェース記述を操作することによって、クライアントは、フレームワーク内のサービスを利用することができる。フレームワークは、操作のための標準的なロケーションを提供するために、それぞれのサービスオブジェクトにプログラムおよびユーザインタフェースを公開する。

【0124】図7に、本発明のアクティブコンフィグレーションモデルを示す。本発明によれば、ネットワーク内のあらゆるデバイスあるいはサービス701は、その機能インタフェースの定義702を指定する。これらのインタフェース定義は、アナウンス705によってクライアントに伝えられる。これらのインタフェース定義は、サーバ側で静的(スタティック)である(CORBAにおけるIDLと同様)。これらのインタフェースは、ネットワーク内のすべてのエンティティ間で共有されるが、いずれのユーザアプリケーションによって操作されることも可能である。ユーザアプリケーションは、サービスインタフェースの使用、および、このようなインタフェースによって提示されるメタデータに対する完全なコントロールを有する。ユーザアプリケーション703によってインタフェースまたはその使用の状態に変更704があれば、リファレンス(参照)706によりデバイスあるいはサービスに反映される。

【0125】したがって、本発明は、任意のネットワークサービスを構築するためのプログラム可能な基盤を提

供する。本発明の一実施例では、エンドユーザアプリケーションは、利用可能な機能および相互作用モードにおける変更に応答して動的に適応され再構成される。この実施例は、コードをダウンロードする Java アプレットの考え方と、標準化されたインタフェースの記述および操作との利点を組み合わせている。本発明のフレームワークでは、ユーザ（あるいはマシンにより生成されたユーザアプリケーション）は、単に、サービスによって提供されるインタフェース記述を編集し、その編集をデバイスに反映させることによって、ネットワークあるいはデバイスの状態を変えることができる。さらに、デバイスあるいはサービスは、標準化された API を通じてアクセス可能である。すべての API を標準化しようとすることによって特徴づけられる従来のシステムとは異なり、本発明のフレームワークによれば、さまざまなネットワークデバイスへのベンダは、自己の製品の記述をフレームワークにマッピングすることが可能である。こうして、ベンダは、構文的定義およびデバイスの能力に集中することができる。

【0126】本発明によれば、デバイス記述は、宣言的スタイルの XML を用いて格納され、アプリケーションコードに追加して使用される。XML デバイス記述の主な機能は、データおよび制御フロー情報を提供することである。この制御/データの分離を公開することにより、本発明は、メタデータを設計に明瞭に組み込むので、格納および操作がアプリケーションとは独立となるため、アプリケーションは、将来の変更が可能のように設計することができる。

【0127】本発明のフレームワークで用いられているような、プログラム/ユーザインタフェースを、それらが参照するオブジェクトから分離することは、当業者に周知の Smalltalk™ のモデル/ビュー/コントローラ (M/V/C) アーキテクチャに類似している。モデル/ビュー/コントローラアーキテクチャについてさらに詳細には、G. Krasner and S. T. Pope, "A Cookbook for Using the Model View Controller User Interface Paradigm in Smalltalk-80", Journal of Object Oriented Programming, August/September 1988, に記載されている。

【0128】M/V/C アーキテクチャでは、データ (モデル) は、データの表示 (ビュー) およびデータを操作するイベント (コントローラ) から分離される。同様に、本発明のシステムにおける文書は、データを操作し見るためのユーザインタフェース/プログラムにそのデータを関連づける糊 (グルー) として作用する。

【0129】本発明は、XML 構文を利用して、デバイス記述スキーマを、XML 文書型定義 (DTD) として作成する。XML は、構造化された情報を含む文書のためのマークアップ言語である。これは、SGML のサブセットであり、階層的な名前付けされた値と、他の文書

を参照するための高度な手段の形で、自己記述型のカスタムマークアップを提供する。XSL (Extensible Style Sheets) や XLink (Extensible Linking Language) のような同類のプロトコルとともに、XML は、付随する文書スキーマ (文書型定義: DTD) のグループを指定し、発見し、結合する能力を提供する。しかし、HTML とは異なり、XML におけるタグのセットはフレキシブルであり、タグのセマンティクスは、その文書に付随する DTD によって定義される。Resource Description Format, Dublin Core および XML-Data のような他のメタデータマークアップ言語も提案されている。

【0130】本発明のフレームワーク内では、デバイスあるいはサービスは、自己の記述スキーマを、XML 文書と、それに伴う DTD およびスタイルシートとして作成する。このスキーマは、言語独立なサービス記述のため、ならびに、ユーザインタフェース (プログラム) をサービスオブジェクトに、およびその逆にマッピングするための、マークアップタグを提供する。また、スキーマは、サービスのインタフェースを XML/XSL 定義内に組み込むので、パーサは、ユーザアプリケーションにコードをダウンロードすることを必要とせずに、これらのインタフェースを読み出すことができる。例えば、電灯スイッチのための <ON> タグは、デバイスがメソッド呼び出しおよびその他のイベントをリスンするアドレスおよびポート番号を示すことが可能である。

【0131】ユーザインタフェースをこれらのサービス記述から生成するために、XML パーサが、クライアントユーザアプリケーションによって使用される。クライアントは、語彙的な型をユーザインタフェースウィジェットにマッピングし、XML/XSL を解析した後、ネットワークの現在のコンフィグレーションに対するユーザインタフェースを生成する。ユーザ (あるいはユーザアプリケーション) は、そのデバイスに対応する動的に生成される UI ウィジェットを単にクリックすることによって、任意のネットワークデバイスの状態を変更することができる。このアクションは、ユーザのマシン上で現在の UI 状態を変更するとともに、適当なコマンド (デバイスベンダにより定義され XML 文書に埋め込まれる) を実行のためにデバイスに送る。本発明の実施例では、この目的のために、Java™ で書かれたパブリックドメインの XML パーサと、Internet Explorer™ 5.0 のような XML 対応ウェブブラウザが使用される。

【0132】以上、本発明について、その好ましい実施例を用いて説明したが、当業者には容易に認識されるように、本発明の技術的範囲および技術思想から離れることなく、形式および細部におけるさまざまな変更が可能である。

【図面の簡単な説明】

【図 1】アクティブコンフィグレーションフレームワークのインフラストラクチャの図である。

【図2】アクティブコンフィグレーションフレームワークの全体設計図である。

【図3】2つのプラグアンドプレイブローカ間の通信を示す図である。

【図4】プラグアンドプレイブローカの内部表現を示す図である。

【図5】サービス発見（ディスカバリ）手続きを示す図である。

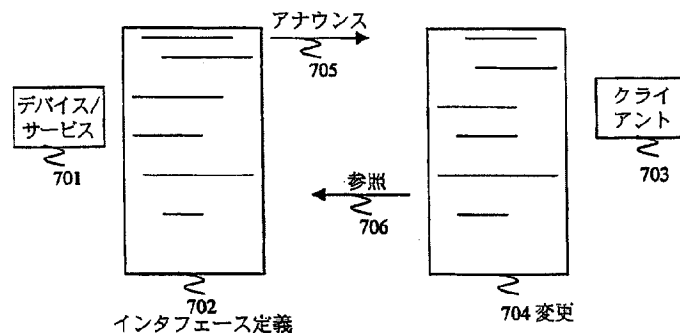
【図6】アクティブコンフィグレーションフレームワークの実装例の図である。

【図7】アクティブコンフィグレーションインタフェースモデルの図である。

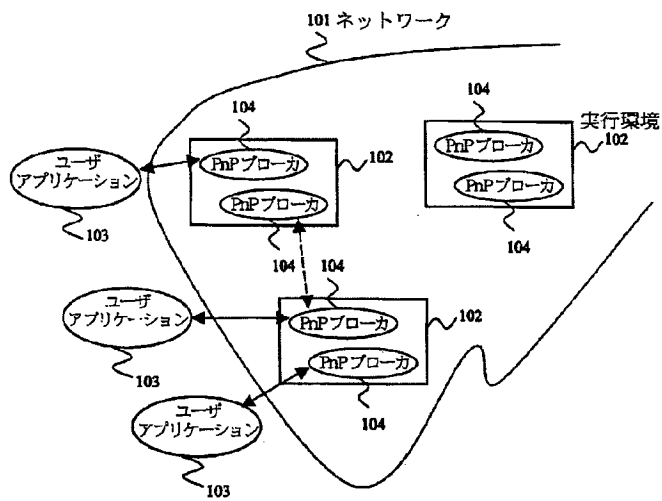
【符号の説明】

- | | | | |
|-----|----------------------------------|------|-------------------|
| 101 | ネットワーク | 503 | PnPブローカ |
| 102 | 実行環境 | 504 | サービスエージェント (SA) |
| 103 | ユーザアプリケーション | 507 | ブロードキャストによるディスカバリ |
| 104 | PnPブローカ | 508 | SLPディレクトリ |
| 201 | ゲートウェイデバイス | 509 | LDAPディレクトリ |
| 202 | ユーザアプリケーション | 510 | 変換スキーマ |
| 203 | ダミーデバイス | 511 | ディレクトリエージェント (DA) |
| 204 | セキュリティ実施機構 | 512 | サービス |
| 205 | ネットワークノード | 601 | テレビジョン |
| 301 | PnPブローカ | 602 | HomePNAクラスタ |
| 302 | PnPブローカインタフェース | 603 | ホームPC |
| 303 | PnPブローカ間プロトコル | 604 | 電話機 |
| 304 | サービスディスカバリ/アベイラビリティエージェント | 605 | レジデンシャルゲートウェイ |
| 305 | サービスレジストリ | 606 | xDSLクラスタ |
| 306 | サービスセッション管理エージェント | 607 | ホームPC |
| 307 | サービスロケーションプロトコル (SLP) サービスエージェント | 608 | ブリッジ |
| 308 | SLPユーザエージェント | 609 | Jiniクラスタ |
| 404 | PnPブローカ間プロトコル | 610 | USBクラスタ |
| 406 | サービスユニット | 611 | スピーカ |
| 501 | PnPブローカ | 612 | ネットワークカメラ |
| 502 | ユーザエージェント (UA) | 613 | X-10クラスタ |
| | | 614 | 白熱電球 |
| | | 615 | セキュリティマネージャ |
| | | 616 | イーサネットクラスタ |
| | | 617 | デジタルカメラ |
| | | 618 | ウェブパネル |
| | | 619 | イーサネットハブ |
| | | 620 | ホームPC |
| | | 701 | デバイス/サービス |
| | | 702 | インタフェース定義 |
| | | 703 | クライアント |
| | | 704 | 変更 |
| | | 705 | アナウンス |
| | | 706 | 参照 |
| | | 1104 | リモートPnPブローカ |

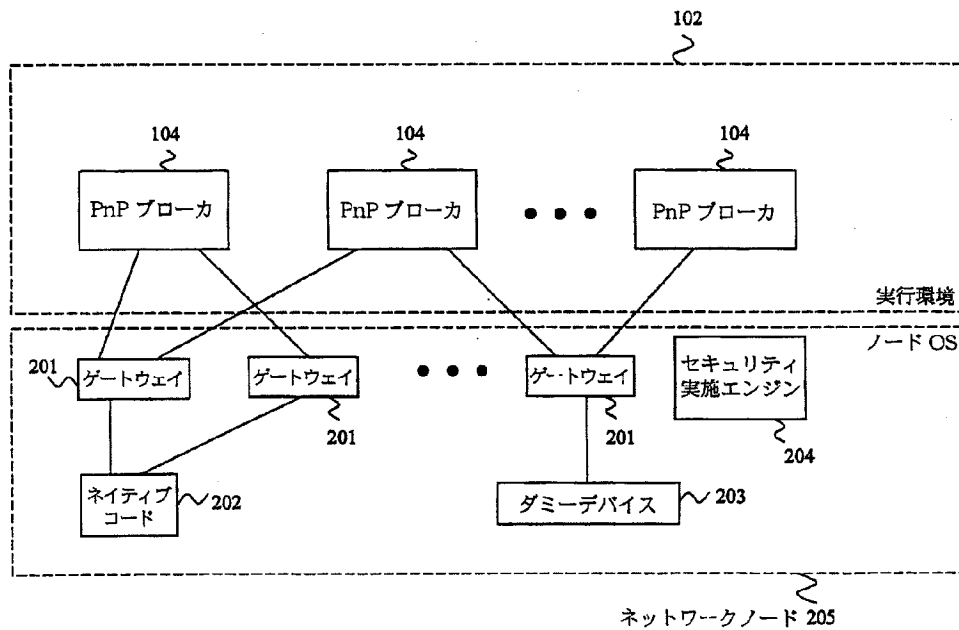
【図7】



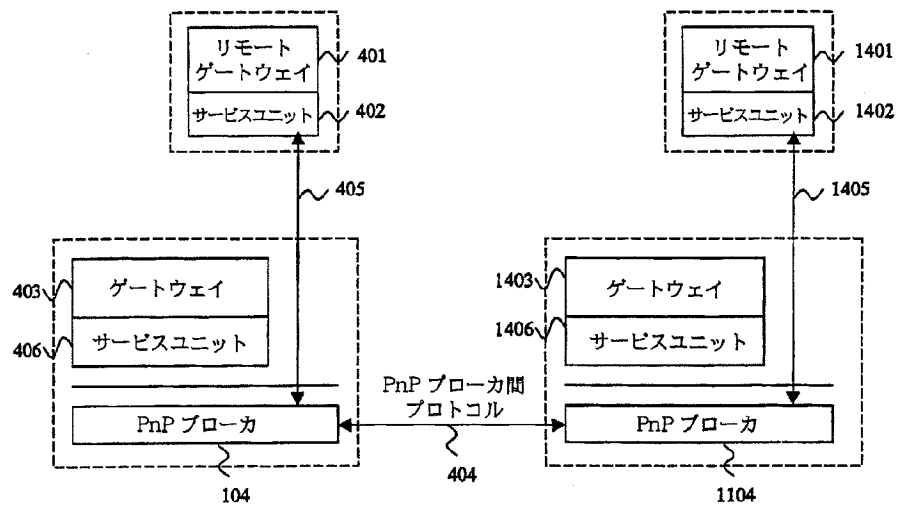
【図1】



【図2】



【図3】



【図4】

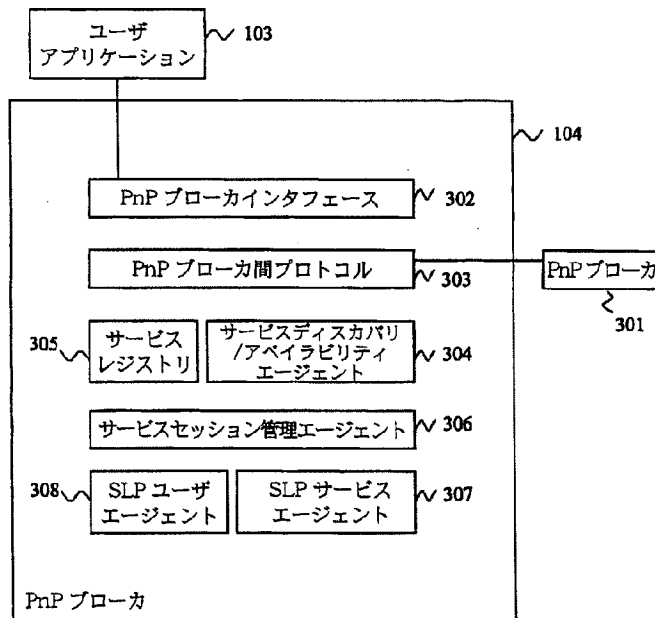


Figure 1 illustrates the system architecture. The components and their interactions are as follows:

- 103 ユーザアプリケーション (User Application)** interacts with the **PnP ブローカ インタフェース (PnP Broker Interface)** via **検索問合せ (Search Request)**.
- The **PnP ブローカ インタフェース** connects to the **501 PnP ブローカ (PnP Broker)**.
- The **501 PnP ブローカ** contains the **502 ユーザエージェント (UA) (User Agent)**.
- The **508 SLP ディレクトリ (SLP Directory)** is a central directory component.
- The **509 LDAP ディレクトリ (LDAP Directory)** interacts with the **508 SLP ディレクトリ** via **LDAP スキーマ (LDAP Schema)**.
- The **509 LDAP ディレクトリ** also interacts with the **510 スキーマ変換 (Schema Conversion)** component.
- The **510 スキーマ変換** component interacts with the **511 ディレクトリエージェント (DA) (Directory Agent)**.
- The **511 ディレクトリエージェント (DA)** interacts with the **502 ユーザエージェント (UA)** via **505 SrvReq/SrvRply**.
- The **512 サービスユニット (Service Unit)** interacts with the **登録/登録解除 (Registration/Deregistration)** interface.
- The **登録/登録解除** interface connects to the **503 PnP ブローカ インタフェース (PnP Broker Interface)**.
- The **503 PnP ブローカ インタフェース** connects to the **503 PnP ブローカ (PnP Broker)**.
- The **503 PnP ブローカ** contains the **504 サービスエージェント (SA) (Service Agent)**.
- The **504 サービスエージェント (SA)** interacts with the **511 ディレクトリエージェント (DA)** via **506 SrvReg/SrvDereg**.
- The **502 ユーザエージェント (UA)** and **504 サービスエージェント (SA)** both participate in **507 ブロードキャストによるディスカバリ (Discovery by Broadcast)**.

(51) Int. Cl. ⁷
H 0 4 L 29/06

F I
H O 4 L 13/00

テラコト* (参考)